January 2010

Novel Algorithms for Structural Alignment of Non-coding RNAs

Diana Kolbe
Washington University in St. Louis

WASHINGTON UNIVERSITY IN ST. LOUIS

Division of Biology and Biomedical Sciences

Computational Biology

Dissertation Examination Committee:

Sean Eddy, Co-Chair

Gary Stormo, Co-Chair

Michael Brent

Jeremy Buhler

Kathleen Hall

Robi Mitra

Ting Wang

Novel Algorithms for Structural Alignment of Non-coding RNAs

by

Diana Lynn Kolbe

A dissertation presented to the
Graduate School of Arts and Sciences
of Washington University in
partial fulfillment of the
requirements for the degree
of Doctor of Philosophy

August 2010

Saint Louis, Missouri

ABSTRACT OF THE DISSERTATION

Novel Algorithms for Structural Alignment

of Non-coding RNAs

by

Diana Lynn Kolbe

Doctor of Philosophy in Biology and Biomedical Sciences

(Computational Biology)

Washington University in St. Louis, 2010

Sean Eddy and Gary Stormo, Co-Chairmen

Non-coding RNAs are biologically important molecules, with a variety of catalytic and regulatory activities mediated by their secondary and tertiary structures. Base-pairing interactions, particularly at the secondary structure level, are an important tool for identifying and studying these structural RNAs, but also present some unique challenges for sequence analysis. Probabilistic covariance models are effective representations of structural RNAs, with generally high sensitivity and specificity but slow computational speed.

New algorithms for dealing with structural RNAs are developed to address some of the practical deficiencies of covariance models. A new model of local alignment improves accuracy for fragmentary data, such as found in direct shotgun sequencing and metagenomic surveys. A separate and alternative model of local alignment is used as the basis for a structural search filter. This is combined with other filtering techniques and high-performance implementations to increase the practical speed of high-sensitivity search. As a whole, these improvements provide a foundation for and point toward future improvements in noncoding RNA homology search.

Acknowledgments

I have received substantial financial support over the course of my doctoral candidacy; the National Science Foundation's Graduate Research Fellowship Program and Howard Hughes Medical Institute have my gratitude.

In addition to Sean, who as an advisor has always been confident in my abilities and allowed me to pick challenging projects, the members of the Eddy lab past and present have been a pleasure to work with. Without Goran's peerless computer support, much of my work would not have been possible; without my various coordinators Melanie, Mary, and Margaret, I might still be mired in paperwork. Thanks to Elena and Eric for many productive conversations, but also to Jenny, Tom, Robin, Sergi, and Fred for interesting topics, perspective from outside RNA sequence analysis, and good humor.

Finally, I'd like to thank my parents, and especially Dennis, for being unfailingly supportive and encouraging, even when they had no idea what I was talking about.

Contents

List of Figures

Chapter 1

Introduction

1.1 Scope

The work I describe here includes two major projects and a few smaller offshoots, but each of them at its core revolves around a particular answer to the question:

What is the meaning of 'local alignment' in the context of sequences with secondary structure?

Without secondary structure, local alignment is commonly understood to consist of the alignment of continuous subsequences - adjacent sets of residues in the one linear dimension of each sequence. The introduction of secondary structure to a sequence or alignment is, in essence, an additional dimension or axis for evaluation. This raises the possibility that a local alignment could be a set of adjacent structural elements, such as the base pairs in a helix. Thus, local alignment including secondary structure must specify whether the atoms for alignment are residues, structure elements, or some combination of both.

The choice of alignment type must reflect the properties of the data that is being considered. The trCYK algorithm, described in chapter 2, explores the case in which a local alignment is contiguous with respect to the sequence, but discontinuous with respect to secondary structure. This method is most applicable to cases where only incomplete sequence data is available. The alternative approach, in which aligned structures are complete but their corresponding sequence is disconnected, forms the basis for the structural search filter

in chapter 5. In this case, the alignment method is designed to recognize the conserved core of structural elements while leaving the complete alignment of more divergent regions for later steps.

The remainder of the work deals with the more technical questions which arise after the design of a new algorithm. What new parameters does the model use, and how should they be set? (Chapters 2 and 5.) What are the score distributions of the alignments, and can we estimate statistical significance given a score? (Chapters 3 and 5.) How long does the algorithm take, in concrete and relative terms, and how does is scale to larger problem sizes? (Chapters 2 and 5.) How does my approach compare in performance to alternative approaches that already exist? (Chapter 5.)

Before getting to those questions, the remainder of this chapter will be a brief overview of some of the foundational areas for this work, concentrating mainly on sequence alignment, non-coding RNAs, and the intersection of the two.

1.2 Sequence search and alignment

The study of biological sequences – DNA, RNA, and proteins – is fundamentally a comparative exercise. Although full understanding must come from careful experimental work, we learn more by considering related sequences as families rather than isolated examples, and comparative data can direct experiment by highlighting similarities and differences. These comparisons rely on a frame of reference or correspondence between the sequences, an annotation indicating the likely evolutionary relationships between the various portions of each sequence. Thus, one of the core problems of biological sequence analysis is the alignment of sequences, and its corollary, the searching of sequence databases for sequences which produce good alignments to a query.

1.2.1 Sequence annotation via homology

By aligning sequences and scoring those alignments, we hope to determine the evolutionary relatedness of those sequences, and describe their similarity in a quantitative way. A 'significant' alignment is used to infer

homology, implying a common origin of the sequences, with the level of similarity or difference hinting at a relative distance in evolutionary history. Formally, many alignment and scoring methods measure sequence similarity, rather than strict homology; however, similarity is a useful (and more tractable) proxy.

This inference of homology is used in many ways: to transfer functional information from a sequence with annotation to one without; to build phylogenies of the species from which the sequences are derived; and to examine rates of change within the sequences to determine which portions are most constrained by evolution, among others. Because this is such a central problem in sequence analysis, the quality of the alignments is of great importance; a poor alignment can confound any further analysis.

1.2.2 Sequence to sequence alignment

An alignment is a mapping of the relationships between positions in two (or more) sequences. It can be represented in many ways; the most familiar is probably the row-column style, where each sequence is written out on its own line, possibly with spacers inserted, such that residues which are in the same column are inferred to be homologous in origin, analogous in function, or both, depending on the nature of the alignment. The spacers required to pad out a sequence to the appropriate length are known as gaps, and represent evolutionary insertion or deletion events. Unless we have or can infer the ancestral sequence, we cannot generally distinguish between insertions and deletions – an insertion in one sequence will produce the same alignment as an equivalent deletion in the other. Ungapped alignments, allowing only substitution events, are also possible, and are sometimes used for very short motifs, such as transcription factor binding sites. However, gapped alignments represent the more general case, and most alignments will at least allow the possibility of gaps, even if they are strongly disfavored.

1.2.2.1 Global alignment

The most basic, and earliest developed, methods for sequence alignment consider exactly two sequences at a time. The algorithm of Needleman and Wunsch [189], along with later refinements by Sellers [220] and Gotoh [92], searches for the best possible alignment of two sequences (given a scoring scheme, described in

1.2.2.3). This algorithm aligns the entire length of both sequences, which is called 'global alignment'.

The Needleman-Wunsch algorithm, along with many of the others which will be introduced later, is from a class known as 'dynamic programming' or DP. This type of algorithm uses a recursive formulation of the problem in order to exhaustively search the solution space without explicitly enumerating all possible solutions. In this case, the algorithm recursively defines the best possible alignment of two subsequences, such that any such alignment may be scored as the sum of the last alignment column, and the best alignment of the remaining subsequences. The complete alignment can then be built by starting with the shortest possible alignments (a single column) and extending them one column at a time. By starting with the smallest alignments and carefully planning the order in which alignments are considered, it is possible to ensure that the score for the current alignment under consideration is simply a marginal sum to a value which has already been calculated and stored.

The complexity of these algorithms is frequently expressed in 'Big-O' notation, which gives the worst case behavior of the method as a function of an abstract measure of the problem size. This type of sequence alignment has time complexity $O(MN)$, where M and N represent the lengths of the respective sequences. Intuitively, this means that doubling the length of one sequence will double the time required, and doubling the length of both sequences will quadruple the time. Often, this will be further simplified to $O(N^2)$, with N being the size of the longer sequence, since M and N are usually near the same size.

1.2.2.2 Local alignment

Global alignment includes the entire lengths of the input sequences, which represents an assumption that the sequences are in fact homologous over their entire lengths. More commonly, though, biological sequences may only be detectably similar over a portion of their length. For example, proteins may share a common domain but have others which are from different origins, or a sequence may be sufficiently diverged such that only the portions with the highest degree of evolutionary constraint are recognizably similar.

For these reasons, we frequently wish to search for the best alignment of portions of the input sequences. Local alignment is exactly that, finding the pair of subsequences with the maximum alignment score. Like

global alignment, this can be described in a dynamic programming framework guaranteed to find the optimal solution, and this was first done by Smith and Waterman [224].

1.2.2.3 Substitution matrices and gap penalties

To this point, I have assumed the existence of some method to evaluate or score an alignment. The assumptions of the alignment method dictate certain properties of the scoring scheme. Namely, since the alignment is built via the addition of independent columns, the score must also be calculated as a sum of independent columns. Columns in pairwise alignment may logically be divided into two classes: those where two residues are aligned to each other, and those where a residue in one sequence is aligned to a gap in the other, representing an insertion or deletion event.

When considering columns of aligned residues, intuitively we wish to give high scores to residues which are similar, and likely to preserve function, and low scores to residues which are more different. The simplest scoring systems award a positive score for matching identical residues, and a negative score for all substitutions. These match/mismatch scores (such as +5 for match and -4 for mismatch) are typically used for alignments of DNA, and represent the assumption that all substitutions are equally likely. More commonly, and especially for proteins, there is an all-by-all matrix of substitution scores, reflecting their observed occurrences in trusted alignments of known homologs, as in the PAM [50] and BLOSUM [112] matrices.

These scores are usually expressed as log likelihood ratios, or 'log odds', measuring the relative probability of an aligned pair of residues in alignments of homologs ($p_a b$) compared to their frequencies in random sequences (f_a, f_b) as given by

$$s(a,b) = \log\left(\frac{p_{ab}}{f_a f_b}\right)$$

These scores have the attractive property that pairs that are more common in real alignments than expected by chance will have positive scores, and those that are seen less often than predicted by chance will have negative scores.

Although it is possible and sometimes useful to consider gapless alignments, for most applications it is preferable to allow gaps for insertions and deletions of sequence residues. Gaps in an alignment are typically

penalized with negative scores, either with linear cost (equal penalty for each gap), or with an affine cost (a larger penalty for the first gap, and smaller penalties for extending an existing gap to additional columns). The original Needleman-Wunsch alignment algorithm [189] allowed arbitrary gap penalties depending on the overall length of the gap; however, this is computationally expensive, and subsequent methods have generally simplified the problem to an affine gap model [92].

1.2.3 Sequence to profile alignment

Frequently, we have not one or two but many examples of a particular family or class of sequences, and by analyzing them as a group rather than one by one we can extract information about conserved features and characteristics. Profile-based analysis generally involves two major phases: first is the creation of a multiple sequence alignment (MSA) that captures the known information about the relatedness of a family of sequences; this is followed by application of that alignment or model to potential new members of the same family.

1.2.3.1 Multiple sequence alignments

Intuitively, it is simple to extend the concept of a pairwise alignment to multiple sequences: columns still represent homologous residues, and there are as many rows as there are sequences. Unlike pairwise alignment, there are not feasible polynomial-time solutions for exact alignment of arbitrary numbers of sequences. As such, dynamic programming approaches tend to be impractically time-consuming for examples beyond a small number of sequences [164], and heuristic approaches are employed instead. Algorithms for multiple sequence alignment generally rely on combinations of hierarchical clustering, stepwise progressive alignment, and iterative refinement techniques [28, 71, 191, 240]. Generally speaking, alignment starts with all pairwise comparisons of the sequences, calculating a phylogenetic tree if one is not already known, and then proceeds with pairwise alignment of the two most closely related sequences. This alignment is then a new 'sequence' which replaces its two component sequences in the tree, and the procedure is repeated until all sequences have been added. Because this algorithm is greedy, fixing the partial alignment at each step, it is

often necessary to refine the alignment by partitioning the sequences of the alignment and re-merging the two groups, until some convergence criteria is reached.

Alternatively, an alignment may be built by iterative application of the profile search techniques discussed in the next sections, starting from a seed potentially as small as a single sequence [7, 129]. A full description of the methods for creating multiple alignments is beyond the scope of this work, and has been extensively covered in reviews [72, 93, 190, 249].

1.2.3.2 Position-specific weight matrices

The substitution matrices used for general sequence alignment are typically constant over every column of an alignment; for example, any A aligned to any G will always receive the same score. However, given a multiple sequence alignment, we potentially have much more information about the selective pressures on a particular sequence position. The same substitution could receive a much lower score in a column with high conservation compared to one with more frequent mutations. Position-specific weight matrices (also known as position-specific scoring matrices, or PSSMs) use this variation to build a set of residue scores for each position in the alignment. New sequences can then be evaluated based on how well they score, i.e., how well they match the observed pattern. When PSSMs are used as ungapped models, they tend to be used for short motifs such as transcription factor binding sites [177, 215, 235, 246]. If these models are used for gapped alignment, amenable to larger sequence features, the gaps are usually scored with *ad hoc* gap open/gap extend penalties like those seen in pairwise alignment [7, 172], and these gap scores can be adjusted to be position-specific, as in the profile tools of Gribskov et. al. [97, 98].

1.2.3.3 Hidden Markov models

Profile hidden Markov models are similar to PSSMs in that they also use position-specific scores, but they have the important distinction of being fully probabilistic models, explicitly modeling column positions as states with emission probabilities, and transition probabilities controlling the path between these states. This means that insertions and deletions can now be modeled as probabilistic events (which are also position-

specific), and importantly, that scores can be directly interpreted as the likelihood of a particular sequence or alignment under that model. These features give hidden Markov models (HMMs) a great deal of flexibility for modeling longer sequence features with high variability, and they have been extensively used in areas such as protein similarity search [63, 67, 149], with several databases devoted just to HMMs of protein domains [78, 159, 259]. Not all HMMs are profile models; applications in other aspects of sequence analysis include genefinding, where the data being modeled is not a profile of a particular alignment, but an aggregate preserving features such as codon structure and intron-exon boundary markers [30, 111, 146, 150, 231].

There are known, efficient algorithms for application of HMMs; these can generally be broken into the classes of training algorithms, for estimating the parameters of the model, and alignment or decoding algorithms, which interpret data as having been produced by the model. HMM alignment algorithms are quite similar to the earlier foundation of sequence alignment; rather than aligning two sequences to each other, one sequence is replaced with a series of model states which represent the family. The alignment algorithms calculate either the most likely state path that could have produced the observed data, along with the combined probability of that path and the data, in the case of the Viterbi algorithm [245], or they calculate the probability that the data was produced by the model, over all possible paths, in the case of the Forward algorithm [63, 201]. In the latter case, with the addition of the analogous Backward algorithm, the probabilities can be further analyzed to determine the posterior probability that a particular residue was associated with any of the model states, which is known as posterior decoding.

For learning model parameters, if the state paths of the training data are known, it is simple to produce maximum likelihood estimates of the probabilities. For profile-HMMs, the state path can be considered known if there is an existing MSA (rather than a set of unaligned sequences), and if there is a simple rule (or manual annotation) specifying which of the columns of the alignment should be considered 'consensus' features of the sequence family. However, if the alignment and therefore the state correspondence is unknown, parameter training then generally takes the form of Expectation-Maximization (EM) [52], iteratively determining the parameters of the model and refining the resulting alignment until some convergence criteria is met and the parameters have reached a local optimum. Often, EM maximizes the Forward likelihood,

which is a special case known as Baum-Welch optimization [18]. Alternatively, training can optimize the likelihood of the Viterbi path. This is only a brief summary of the capabilities of HMMs; for more detail, consult Rabiner [201] for a general overview, or Durbin [63] for application to biological sequences.

1.2.4 Incorporating additional information

So far, I have discussed alignment based only on primary sequence information, considering each residue in the sequence independently of all other residues. However, a functional sequence is often more than the sum of its parts. Residues may interact with each other such that it is the combinations of residues rather than individual positions that are informative. If these interactions are predictable, we can design models which capture the additional information, and produce alignments that take those features into account.

1.2.4.1 Secondary and tertiary structure of RNAs

RNA transcripts *in vivo* are not typically found as stretched out and unwound strands. Instead, they tend to fold into more stable and energetically favorable structures utilizing the Watson-Crick basepairs of DNA as well as a variety of other hydrogen-bonding interactions. This is especially true of RNAs that are not protein-coding messengers, where the function of the molecule may be directly mediated by its folded, three-dimensional structure. Therefore, when aligning non-coding RNAs we may wish to consider not only whether the alignment is reasonable in terms of sequence conservation, but also whether the sequences are compatible with an inferred or explicit shared structure. For instance, a simple approach might require that positions aligned to an annotated base pair must also be able to form a base pair. This type of consideration is not limited to canonical base pairs. Other interactions such as conserved non-canonical pairs, base-triples, or larger units such as tetraloops are also possible targets for structural alignment. At a higher level, tertiary interactions affect the packing of loops and helices, and control which portions of the sequence are exposed for interactions with other molecules.

Generally, current approaches to non-coding RNA concentrate on secondary structure, and incorporate base-pairs (canonical or otherwise) and loop models in addition to primary sequence. I will discuss RNA

structure and the approaches to incorporating it into alignment in more detail, beginning in section 1.3.

1.2.4.2 Secondary and tertiary structure of proteins

Higher order structure can also be important in detecting similarity between protein sequences, although it is seldom used for direct sequence alignment or searching. Patterns of the secondary structure elements alpha-helices, beta-strands, and beta-sheets are often used to classify protein families into classes and folds, even when primary sequence similarity is no longer evident [183]. These secondary structure elements can be used as coarser-grained units for alignment, either when measuring similarity between known structures [148], or when they have been predicted *de novo* or via alignment to be used for tertiary structure prediction via threading [25, 130, 226]. Typically, structural alignment of proteins is only used when explicitly trying to model the three-dimensional structure of an unknown sequence, a large field tangentially related to sequence alignment. For recent reviews of protein structure prediction techniques, see the evaluations of the Critical Assessment of protein Structure Prediction (CASP) competitions [151, 181, 182, 274], and for an overview of protein structure-to-structure alignment, see Hasegawa and Holm [106].

1.3 Structural RNAs

There is tremendous variety in the nature and role of non-coding RNA elements (ncRNA) [65, 125]. RNAs are known to function directly in many of the central activities of the cell, including transcription, translation, and gene regulation. These RNA elements generally function through folding to produce a structure (intramolecular interactions), through base-pairing with a nucleic acid target or binding with a protein (intermolecular interactions), or through some combination of the two. For example, H/ACA snoRNAs, which guide base modification of ribosomal RNAs, have a conserved fold and structure, but also have sites that direct target recognition [84]. Although some ncRNAs are unstructured and function exclusively through binding to a target, the methods that I will be discussing are designed to incorporate information about conserved secondary structure. Therefore, I will limit myself to considering only those ncRNAs that function at least partially via a conserved intramolecular fold, i.e., structural RNAs.

1.3.1 Structural elements in RNA

Base pairs, stacks, and helices

The canonical Watson-Crick base pairs are the most common secondary structure interaction in RNAs; most other structure elements involve two or more base pairs. In addition to the standard A-U and G-C pairs, RNA also frequently uses the G-U wobble pair, which binds with only slightly less strength than an A-U pair [175].

Base pairs are not typically found in isolation; because the bases can be oriented in parallel planes, two adjacent base pairs can form a stack. By excluding water, this structure is more stable than the two pairs in isolation, and any additional base pairs that are added also benefit. As several base pairs stack, the residues spiral around a central axis, forming a helix very similar to the A-form structure of DNA, although distinct in its precise geometry [119, 199].

Less commonly, but also importantly, RNA also exhibits base-base interactions involving the sugar and Hoogsteen edges. These pairs may use bases not typically thought of as pairing, such as A-C and A-G; however, these interactions are observed both as conserved and covarying in sequence analysis, and are clearly present in high-resolution structures. Because of their unusual geometry, these interactions are somewhat less likely to be within stacked helices.

Hairpin loops

A hairpin loop refers to a (frequently short) stretch of unpaired residues that directly connect the two sides of a helix, with the sequence doubling back on itself. (By extension, a hairpin consists of both the loop and the helix.) Some hairpin loops are large and flexible, but many hairpin loops are just 4 nucleotides long ('tetraloops') and compactly folded, forming a tight turn [39, 114].

Bulges, internal loops, and multi-loops

All other unpaired sequence can be divided into two groups based on whether or not it is enclosed by a base pair. If it is not, then it would be considered a simple single-stranded region. Unpaired nucleotides that are enclosed by base pairs, on the other hand, are all considered some type of loop structure. Bulges describe

when there is one or more unpaired residue inserted in a helix, but only on one side; if there are residues on both sides, then it is considered an internal loop. Multi-loops are the branching junctions where three or more helices converge. Representative examples of each of these are shown in figure 1.1.

Pseudoknots

Pseudoknots are formed when residues that in linear sequence are between the two halves of a helix form their own base pairs with residues that are positioned outside of the helix. This is also called a 'crossing' interaction, since the base pairs cannot be described as a nested structure. Examples of simple pseudoknots are given in figure 1.2. These structures can easily be formed if two hairpin loops interact ('kissing hairpins'), or if a bulge or internal loop interacts with any other unpaired sequence.

Dangles, base triples, and everything else

The structures mentioned above cover only the most basic set of structural elements; there are others, with varying levels of complexity. Some of these are important to consider based on the problem being solved. For instance, thermodynamic folding algorithms have descriptions for 'dangles', single stranded bases that stack on the end of a base-pair helix, along with a much more extensive classification of bulge and loop structures.

Base triples, while far less common than pairs, have an important role in stabilizing the tertiary structure of large RNAs. Although triples were identified as important RNA structures in early structural analysis [161], and some co-varying triples were recognized based on comparative sequence analysis [86, 179], the frequency and variety of forms of triples has only been recognized in the era of large crystallographic structures [158].

Although these (and other) elements can be important for understanding the structure and function of an RNA, they represent a level of complexity that is not easily accommodated in the covariance models on which the majority of this work focuses. For that reason, I will generally restrict myself to the simpler classification set of pairs, helices, and loops.

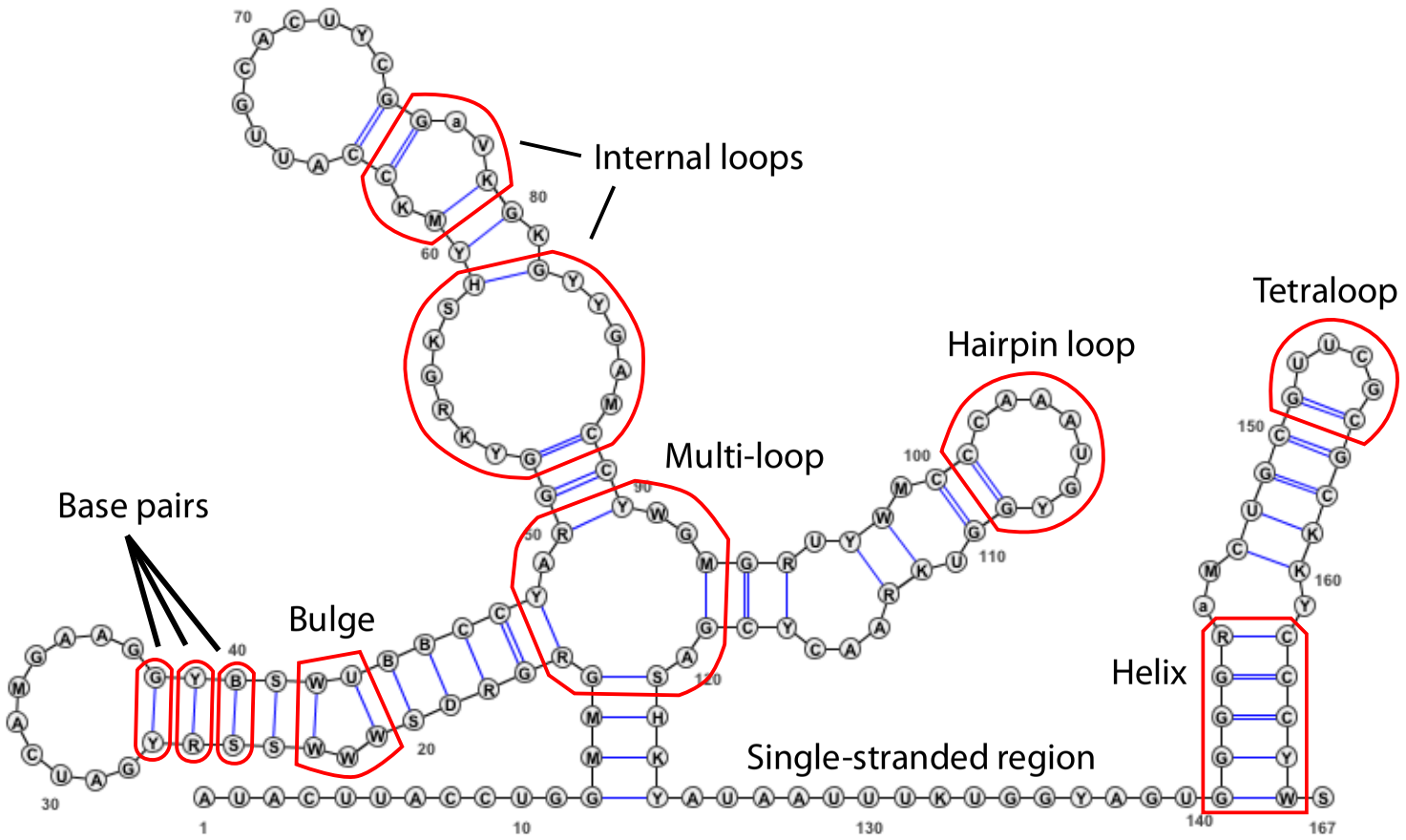


Figure 1.1: **Secondary structure units.** A basic classification of secondary structure units, illustrated on the consensus structure of the metazoan U1 spliceosomal RNA. Structure is from Rfam [85]; secondary structure rendering by VARNA [47]. Residues include IUPAC nucleotide ambiguity codes.

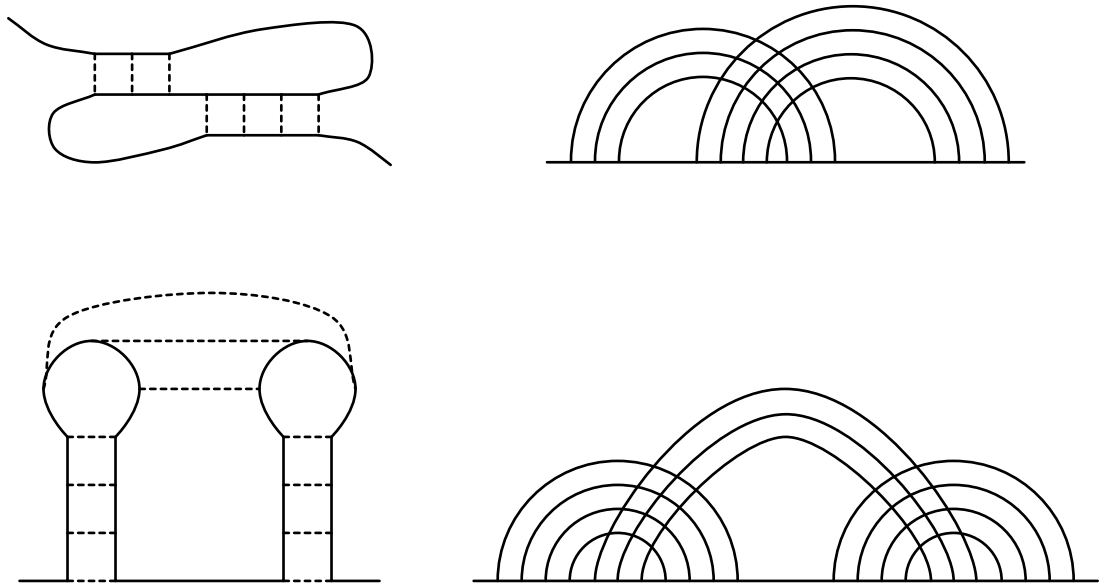


Figure 1.2: **Simple pseudoknots.** At left, secondary structure diagrams for two common types of pseudoknots, the simple pseudoknot (top) and kissing hairpins (bottom). Solid lines indicate the RNA backbone and dotted lines are base-pairing interactions. At right, the same two structures represented as arc-figures: the sequence is stretched out as the horizontal line, base pairs are the arcs above the line. Pseudoknotted interactions produce crossing arcs, while nested structures do not.

1.3.2 Biological roles and variety

1.3.2.1 RNAs in translation

rRNA The most abundant RNAs in a cell, ribosomal RNAs (rRNAs) are the heart of protein synthesis. The ribosome is formed of a large and a small subunit, each of which has a core of non-coding RNA and a large assembly of proteins. The small subunit typically contains a single large ncRNA (SSU rRNA, also known as 16S in bacteria and archaea or 18S in eukaryotes), while the large subunit consists of one longer ncRNA (LSU rRNA, 23S, or 28S) along with one or two smaller ncRNAs (5S in bacteria and archaea, 5S and 5.8S in eukaryotes). The genes for these rRNAs are usually located together in an operon or gene cluster, and although some organisms maintain only a single copy of each, it is common to have multiple identical or near identical copies of the genes [157, 229]. This helps to support the high levels of transcription needed to renew a large population of ribosomes. Each subunit is also associated with dozens of proteins which stabilize the structure and form interaction sites for other molecules. The active site itself, which synthesizes the protein backbone, consists of RNA, making the ribosome an enzymatic RNA, also known as a ribozyme [233]. SSU rRNA is among the most highly conserved of all genes, and has often been used for phylogenetic analysis of all cellular life due to its ubiquitous presence and slow rate of substitution [89, 207, 257, 264].

tRNA Transfer RNAs (tRNAs) form the link between the triplets of the genetic code and their corresponding amino acids, almost exactly in the form predicted by Francis Crick's adaptor hypothesis in the late 1950s [45] and verified soon after [115, 120, 136]. Nearly all tRNAs share characteristic features, with a central stem-loop that leaves the anticodon available for pairing, two 'side-arm' stem-loops that contain the major features for recognition by tRNA-synthetases, and a final closing stem which ends with the acceptor site where the amino acid is attached [135, 136]. Although there are 64 possible codon triplets, not all of them require dedicated tRNA species because of degenerate or 'wobble' pairing [46]. As an example of a 'typical' bacterial genome, *E. coli* has 86 tRNA genes corresponding to 41 anticodons, whereas the compact genome of *Mycoplasma capricolum* has just 29 genes for 28 anticodons, with much more extensive use of wobble pairings, and some mitochondrial tRNA systems have even fewer anticodons represented [23, 227]. The ex-

act number of tRNA genes varies from organism to organism, but like the rRNAs they are often organized in clusters and may be present in high copy numbers [36].

snoRNAs The small nucleolar RNAs (snoRNAs) are involved in maturation of both rRNAs and tRNAs, providing templates for base modification. These RNAs have some conserved structure elements, but they also contain sequences that recognize their modification targets via base pairing. Typically, C/D box snoRNAs direct ribose methylation and H/ACA snoRNAs guide the conversion of uridine to pseudouridine [12], although there are exceptions to this generality, including some snoRNAs which include features of both classes [48, 126].

RNase P The initial transcripts of tRNAs are somewhat longer than their functional forms; as part of the maturation process the 5' ends are cleaved by the ribonuclease RNase P. Like many of the RNA particles described above, RNase P consists of both a protein and an RNA component, however the RNA is required for catalytic function, and the bacterial RNA is functional *in vitro* even in the absence of the corresponding protein [2, 75, 76, 102]. In eukaryotes, RNase P also seems to be involved earlier in tRNA production, being a factor in efficient Pol III transcription (which includes tRNAs, but also a variety of other ncRNAs) [203].

RNase MRP In eukaryotes, the 18S and 5.8S rRNAs are co-transcribed; the ribonuclease RNase MRP is responsible for cleaving the spacer from the 5' end of 5.8S [42, 169, 219]. Like RNase P, which it is evolutionary related to, MRP is a ribonucleoprotein and a putative ribozyme, although it is not functional without the associated protein components [49, 198]. MRP also has a separate (and earlier discovered) function in the mitochondria, where it cleaves the junction between RNA and DNA in mitochondrial DNA replication [43].

tmRNA Transfer-messenger RNA, or tmRNA, shares some characteristics of both tRNA and mRNA. In bacteria, this structured RNA helps to recover a ribosome when protein synthesis stalls (for example, if the mRNA has lost its stop codon). The tmRNA moves into the active site, providing the next amino acid and

also transferring the ribosome onto its own short open reading frame, which encodes a polypeptide tag that will target the nascent (and likely non-functional) protein for degradation [108, 134, 180].

SRP RNA The signal recognition particle (SRP) is a protein-RNA complex that co-translationally recognizes a signal peptide and directs trans-membrane and secreted proteins to the membrane [17, 59, 251]. The binding of SRP also pauses translation of the protein until it has been associated with a co-translational pore complex, which prevents secreted proteins from being released into the cytosol. The RNA is essential for function of the particle; it is known to stabilize a major component protein, and may also be part of the signal peptide recognition interface [59].

1.3.2.2 RNAs in splicing

snRNAs Splicing of messenger RNAs is mediated by a complex known as a spliceosome, consisting of several small nuclear ribonucleoproteins (snRNPs), each of which includes a highly conserved small nuclear RNA (snRNA). Key functions of these RNAs include recognition of the 5' splice site by U1, and of the branch site by U2. In a minor alternative form of the spliceosome, U11 and U12 recognize the alternative consensus sequences found in U12-type introns. Because of low sequence identity it is not immediately obvious whether U1/U11 and U2/U12 are homologous, but if not, they feature convergent evolution, with strikingly similar structure and function [31, 271]. In addition, U4/6 and U5 form a specific series of RNA-RNA interactions in the stepwise splicing process, displacing U1. (The U12-type spliceosome also uses U5, but replaces U4/U6 with U4atac/U6atac, which are again similarly structured.) By virtue of structural and functional similarities between self-splicing Group II introns and the spliceosome, it is thought likely, although not conclusively determined, that the spliceosome is also a ribozyme [90, 241].

Catalytic introns Not all introns are processed by the familiar mRNA pathway used by nuclear protein-coding genes. Some, including many of those found in structural RNA genes, organellar mRNAs, and rare bacterial introns, fold into an auto-catalytic structure, splicing themselves out of the gene. Group I introns are mobile genetic elements, some of which encode a protein endonuclease that is used to copy the intron

via a DNA double-stranded break repair mechanism. As part of a transcribed RNA, the introns are self-removing, although this function is usually supported by recruited host proteins, and the excised intron may integrate into a new gene via reverse splicing followed by reverse transcription [154]. Group II introns are also mobile elements, but rather than being copied as DNA elements they undergo retrotransposition via an intron-encoded reverse transcriptase. (Again like Group I, not all introns retain the enzyme.) Although the Group II introns are also ribozymes, they generally require accessory proteins *in vivo* [154, 241]. Group I and II introns and nuclear introns utilize similar chemistry, using a two-metal-ion mechanism for transesterification [230, 234, 241] , but Group II is the more similar to nuclear splicing, employing an integral adenosine rather than a free guanosine.

Gene-specific splicing control At least one snoRNA is essential for correct splicing of a particular serotonin receptor gene, rather than the usual targets of rRNA or tRNA base modification [138, 139]. It seems to be processed into smaller units and has different protein associations than typical snoRNAs, and so might also represent an entirely separate class of ncRNAs, distinct but possibly derived from the snoRNAs. This example has led to the suggestion that 'orphan' snoRNAs, with no known targets, may be more widely involved in splicing control, or regulation of alternative splicing, although so far additional verified examples are lacking [19].

1.3.2.3 Self-cleaving ribozymes

In addition to the ones described above, there are other examples of enzymatic RNAs, most of which are self-cleaving ribonucleases. The Hammerhead and Hairpin ribozymes are both examples of structural RNA elements from RNA viroids and satellite viruses in plants; cleavage breaks continuous rolling circle replication of the RNA viral genome into individual units [79, 211]. Examples of Hammerhead-type self cleavage are also known in repetitive satellite elements in animals. The HDV ribozyme performs the same unit-length cleavage function in the human pathogen hepatitis delta virus [20, 153].

A structurally quite different ribozyme, GlmS is located in the 5' untranslated region (UTR) of the glmS gene, which synthesizes glucosamine-6-phosphate. In the presence of this sugar, the ribozyme cleaves the

mRNA, leading to its degradation, but with low concentrations of the sugar the transcript is stable and produces more of the synthetic enzyme, creating a regulatory feedback loop [263]. Because the GlmS ribozyme directly binds the sugar ligand, and performs a regulatory function, it is also considered a riboswitch (see 1.3.2.6).

1.3.2.4 Telomerase RNA

In organisms with linear rather than circular chromosomes, a telomerase enzyme is usually required in order to maintain chromosome length after cell division. This enzyme extends the chromosome by adding additional units of a telomeric repeat pattern, and that pattern is specified by the telomerase RNA [21]. (There are exceptions to every rule; the ends of the linear chromosomes of *Drosophila* are maintained by repeated insertion of retrotransposons rather than telomerase [174].)

Although the RNA is not directly responsible for the enzymatic activity of telomerase, it is a large, structured RNA whose shape is important for its interaction with the protein, and only a small part of the RNA sequence is the repeat template. The function of telomerase and the telomerase RNA are clearly homologous among organisms which use this mechanism to maintain linear chromosomes, but the sequence of the telomerase RNA is highly diverged, making identification difficult. Although a conserved core of structural elements has now been identified [239], the telomerase RNA gene remains unknown in some organisms, including well-studied model systems like *C. elegans* [237].

1.3.2.5 Trans-regulatory RNAs

microRNAs The microRNAs are a very large family of regulatory non-coding RNAs which operate partly through structure and partly through primary sequence complementarity. The genetically encoded form of a microRNA is a long inexact hairpin ($\sim$70 nucleotides); this structure is recognized and cleaved giving a short RNA of 22-23 nt from one side of the longer helix. The mature form regulates specific genes by pairing to target sequences in the 3' UTR of mRNAs, with the double-stranded RNA helix eventually leading to the destruction of the transcript [9, 156]. The microRNA pathway is conserved throughout eukaryotes, and is

especially important in metazoa, where it is part of temporal regulation in development, cell proliferation and death, and lineage differentiation [16]. Under the classification scheme I have established, the precursor RNA would be considered a structural RNA, but the short mature form would not, even though it is a functional, non-coding RNA, due to the lack of an intramolecular secondary structure.

Broad-spectrum regulators Whereas the microRNAs target specific genes (albeit potentially large sets of genes), other RNAs are more general regulators. Under limited-nutrient conditions in *E. coli*, the 6S RNA binds to σ^{70}, an RNA polymerase (RNAP) subunit used under normal growth, and represses its activity, allowing transcription to preferentially use σ^S promoters during stationary phase despite lower availability and efficiency of σ^S-RNAP [253]. In a similar mechanism but different overall effect, in eukaryotes the 7SK RNA is a global negative regulator of transcription, binding and repressing an elongation factor [24]. Under stress conditions 7SK is released from the elongation factor, leading to a general increase in transcription levels. Again in *E. coli*, OxyS RNA is somewhere between the two extremes of general and specific regulators; in response to oxidative stress, it directly represses expression of many genes, including a transcriptional activator and the σ^S subunit of RNAP, cascading to a broader range of effects [8].

1.3.2.6 Cis-regulatory RNAs

Not all non-coding RNA elements are free-standing, independently transcribed genes; many are associated with another transcript, usually protein-coding. In these cases, the RNA often functions as part of the expression control mechanism of the gene.

Riboswitches One of the most direct control mechanisms is a recently discovered class called riboswitches, often found in the 5' UTR of metabolism and biosynthesis genes in bacteria. These elements can fold into two alternative and mutually exclusive structures, depending on the presence or absence of a small molecule ligand, which is directly bound by the RNA [261, 262]. Generally, one structure allows translation of the protein-coding gene, while the other blocks it by sequestering the translation initiation site, or by a more complex mechanism [209, 263]. Although riboswitches have been extensively demonstrated in bacteria, very

few are known in archaea and eukaryotes [37, 209].

Operon leaders Operon leader sequences are similar to riboswitches in that they regulate the production of biosynthetic genes depending on the concentration of a small molecule that is closely related the genes being transcribed. However, rather than directly sensing that ligand, the operon leader sequences incorporate a set of alternative secondary structures and a short open reading frame (ORF) which is enriched for a corresponding amino acid. The concentration of that amino acid then limits the translation speed for the ORF: if it is sufficient, the ribosome moves quickly and the RNA forms a terminator structure which causes the still nearby RNA polymerase to stop transcribing the operon. Conversely, if the appropriate amino acid is scarce, the ribosome moves more slowly, the RNA is able to fold into an anti-terminator structure instead, and the operon genes are transcribed to increase production of the amino acid. Because this mechanism depends on simultaneous transcription and translation, it is not expected in eukaryotes, but no examples have been observed yet in archaea either. Bacterial operon leaders are known for tryptophan, leucine, and histidine, among others [145].

IRE The iron response element (IRE) is a short, conserved hairpin important in regulation of iron metabolism through binding of associated iron response proteins. Examples are known both in 5' UTRs, where it leads to translation repression, and in 3' UTRs in other genes, where it contributes to mRNA stability and thus increases translation. These varied activities create a stable feedback loop, where iron acquisition genes are upregulated and iron sequestration genes are repressed when intracellular iron concentrations are low, and vice versa when concentrations are high [113].

IRES Internal ribosome entry sites (IRESs) provide a structure which is recognized and directly bound by the ribosome to begin translation of a downstream gene, as an alternative to the standard eukaryotic route of 5' cap recognition and scanning. These elements are used extensively by RNA viruses to maintain translation of their own genes while disrupting the cap-based translation mechanism used by the host's genes. In addition, some eukaryotic stress-response genes seem to have co-opted the IRES as part of the host's

survival mechanism [109].

SECIS The selenocysteine insertion sequence (SECIS) element is a translational regulator, but it does not control initiation or expression levels; rather, it changes the nature of the associated protein by causing a UGA stop codon to be translated to a selenocysteine instead. In bacteria, these structures appear proximally after the UGA that they affect, frequently positioning them within the remaining coding sequence [166, 275], but in archaea and eukaryotes they are found in the 3' UTR, where they may affect more than one codon within the protein [248, 260].

1.3.3 Structure determination for noncoding RNAs

1.3.3.1 Comparative sequence analysis

Structural RNAs function through a combination of sequence and structure. When a particular fold or structure is important, but the exact sequence is not, the sequence can diverge through compensatory mutations. As a result, overall nucleotide conservation can be low, but with information remaining in patterns of covariation and conserved structure. As a result, many RNA structures have been determined by comparative sequence analysis.

When performed manually, comparative sequence analysis is an iterative process, starting with the identification of a several structural RNA sequences which are believed to be homologous, either through experimental work, or suggested by primary sequence similarity. The sequences are first arranged into a multiple sequence alignment, and then the columns can be examined in pairs, looking for regions with compensatory changes to each other, that can tentatively be identified as possible base pairs. As structural elements are identified, the alignment can be modified and refined to better reflect a conserved consensus. Because identification of covarying columns requires the truly homologous residues to be in the same columns, the quality of the alignment is important. It can be helpful to identify short, highly conserved stretches of residues to 'anchor' the alignment and reduce the ranges where the alignment is uncertain. Diversity in the selection of sequences can be both a help and a hindrance: closely related sequences will provide more confidence in the

initial alignment, while more distant ones will have more of the informative mutations required to establish a structure. If there are a sufficient number of sequences, it may be sensible to use a progressive strategy, starting with a small group of similar sequences for a good alignment but limited structural information, gradually adding the more distant sequences and refining the structure and alignment at each stage.

Substitutions, particularly compensatory ones, are the most common changes, but by no means the only ones. Small insertions or deletions, which change the exact length of a stem or loop are also seen often, although the frequency is variable both between families and between different positions in a single family. Somewhat less common are large insertions or deletions, which can remove or replace entire structural elements. It can be difficult to determine the structure of a large, uncommon insertion due to the lack of comparative data; these loops are often displayed as being unstructured even though they likely do fold in practice.

Manual comparative sequence analysis is a painstaking operation, requiring significant time and experience with recognizing the patterns of RNA evolution. Although the earliest structures determined by comparative sequence data were done by hand, currently there are many computational tools that can assist in the procedure (such as searching for small common structural elements). Other approaches attempt to automate the process entirely, using measures such as mutual information and thermodynamic stability to evaluate large populations of possible structure-alignment combinations. Many of these methods will be described in brief in section 1.3.4.

1.3.3.2 Experimental structure determination

Indirect structural probing Several assays of RNA secondary structure operate in a similar framework, including chemical and enzymatic methods. In general, these assays are conducted with the RNA under native conditions, and modification or cleavage is directed at a subset of bases or conformations. The reactions are calibrated so that on average, a single molecule does not have more than a few modifications or cleavages. This produces RNA fragments of different lengths, which can be amplified and mapped by length, indicating the position of residues which were subject to the reaction. In some chemical probing assays, a small reactive

molecule that modifies one or more nucleotides is introduced to a purified ncRNA, but it can only access those bases that are unpaired in the structure. When this population is amplified by RT-PCR, it results in a variety of sequence endpoints, as the reverse transcriptase cannot continue past the modified base. In other assays, the reactive molecule cleaves the phosphate backbone, also producing a population with a variety of lengths corresponding to the cleavage locations. The enzymatic assays use RNA nucleases which also cleave the backbone, and they are available with a variety of specificities, whether for single-stranded or double-stranded regions, and particular bases or not [73]. A similar assay using hydroxy radicals can also be used to determine regions of the RNA backbone that are protected by protein binding, if higher-order structure is also being investigated. Although these assays do not directly indicate which positions are paired with each other, structural probing can dramatically reduce the number of possible folds to consider, and can be used in conjunction with comparative sequence analysis.

In-line probing operates on a similar principle, but rather than introducing chemical modifications or nucleases, it depends on the natural decay of the RNA backbone to introduce breaks into the molecule. This differentiates between structured and single-stranded regions, because the flexible, unstructured portions of the molecule are more prone to random breakage. In-line probing assays may need to incubate for up to several days to produce enough breakage products to visualize on a gel, but this technique has been used as a functional assay for conformational changes in the presence or absence of a ligand [228, 261].

X-ray crystallography The largest 3-dimensional structures come from X-ray crystallography, which has lately produced detailed structures for a self-splicing intron [241], hammerhead ribozymes [173], and very large RNA-protein complexes, including SRP [17] and several ribosomes [15, 221, 272]. It was these detailed structures which demonstrated an active site completely composed of RNA that conclusively established the ribosome as a ribozyme, a key change in our understanding of the catalytic capabilities of RNA. The caveats of crystallography include concern that crystal packing may induce non-native structures, and loop regions which are flexible often remain unresolved in the structure. However, the largest obstacle to widespread structure determination by crystallography is the sheer difficulty of the endeavor, limiting the number of

structures which can be produced.

NMR spectroscopy Although structures from crystallography have received much attention, important structural discoveries have come from nuclear magnetic resonance spectroscopy (NMR) as well. NMR has different strengths – it hasn't been scaled up to handle very large complexes, but it does allow structure determination in solution, including flexible regions that may interconvert between different stable conformations. As such, its capabilities are largely complementary to crystallography's weaknesses, and so a more complete picture of RNA conformation is gained by using a mixture of both techniques. NMR studies have been used extensively in investigating characteristic tetraloop hairpin turns and their receptors [33, 34, 39, 114].

1.3.4 Computational methods

There is a broad range of methods and applications for computational analysis of structural RNAs. These approaches can be broadly categorized based on a few basic criteria. First is the problem(s) that they try to address, typically recognition, alignment, folding, or some combination of these. Next is the type of input data the program expects, whether from the user directly, or from the designer of the program in terms of the domain-specific knowledge that is encoded in the model. This is closely related to the generality of the approach – approaches which specialize in a particular family of structural RNAs will naturally incorporate much more information about the properties of that family than would a program which is intended for general use. Finally, we sometimes wish to characterize the mathematical underpinnings of an approach: whether the output is a probability or an *ad hoc* quality score; whether the solution is approximate or guaranteed to be optimal; and what are the potential errors, and the error rates, associated with the method.

1.3.4.1 *De novo* folding

De novo folding attempts to assign a secondary structure to a sequence, and is also known as secondary structure prediction. Generally this means finding the structure that maximizes some quality measure. For instance, a very simple folding algorithm proposed by Nussinov maximizes the number of base pairs in the

structure [193]. Although this method is too simplistic to be generally useful, it has an efficient dynamic programming solution which is similar to those used by more complex models.

Usually, secondary structure prediction operates under an explicit thermodynamic model of conformational stability, looking for the minimum free energy structure (MFE) [276, 278]. The parameters of these models are set based on experimentally determined melting energies of various component RNA structures [81, 175, 243, 250]. More recently, there has been a move away from determining only the MFE to considering a population of suboptimal structures [277], or centroid estimators to capture highly populated areas of the folding space, even if none of them individually are the lowest energy structure [54, 55, 194]. Other alternatives have looked at replacing the thermodynamic parameters with information about the evolutionarily conserved folds of RNA families with known structures [57, 60], or incorporating both sources of information [10]. Although structure prediction has generally considered only a single sequence at a time, it is also possible to jointly fold molecules that interact with each other [11]. The algorithms used in secondary structure prediction can produce either exact optimal solutions or statistically sample from the solution space, but they are of course subject to the limitations of their energy model, including some parameters which are estimates rather than experimental measurements. Also, there are the usual caveats about whether an energetically stable conformation corresponds to a biologically relevant one, where a less favorable conformation may be stabilized by its interaction with other molecules.

Thermodynamic folding can also be extended to determine aggregate properties of the entire population of a particular RNA, such as in the McCaskill algorithm, which calculates the overall probability of whether or not certain positions are base paired [178].

1.3.4.2 Folding of aligned RNAs

Similarly to folding a single sequence, it is also possible to fold a group of sequences, looking for a shared structure. In addition to the thermodynamic parameters that are available for the single-sequence case, starting from aligned sequences provides additional information in the form of covariation in mutations. For instance, if two alignment columns have poor sequence conservation but always preserve a canonical base pair, that

is a strong indication that the two positions are paired in the functional structure. Mutual information is a statistical measure of the covariation between two variables, and it is possible to fold aligned RNAs by simply finding the structure which optimizes the sum of the mutual information for columns paired in the consensus structure [40, 104]. However, most modern algorithms in this class use a combination of terms incorporating both thermodynamic stability and covariation information. Alifold uses a linear combination of average thermodynamic energy and a covariation term which is based on a sum-of-pairs calculation for residue changes that preserve base pairs [118]. Pfold uses a more probabilistic approach, including phylogenetic analysis of columns and a folding grammar trained on known alignments of tRNA and rRNA [142, 143].

1.3.4.3 Alignment of sequences with known structures

If sequences have been assigned known structures, we may wish to take those structures into account when performing alignment of those sequences. This problem has not been as extensively explored as some other classes of alignment, as usually only one structure is known, and it is imposed on the second sequence via the alignment. That case is more closely related to the profile-based methods for aligning new sequences to a sequence with a known structure, and will be discussed later (see 1.3.4.6).

These approaches must generally define an objective function to describe the goodness of an alignment of structures along with a method for scoring the primary sequence component, and a weighting determining the relative importance of each. One such method is known as an edit distance, which counts the minimum number of events which must occur to transform one sequence-structure pair into the other [160]. The definition and weighting of the individual events has a strong influence on the types of alignments that are produced by the method; for instance a single base mutation may be considered as two events if the substitution also breaks a base pair. A general edit distance was developed by Jiang et. al. covering the cases of no structure (primary sequence only), and secondary structure, both nested and pseudo-knotted (crossing) [128]. This distance formulation has been adopted by other alignment approaches, such as RNA-forester [116].

Alignment of sequences with known structures also has a local variant, namely, finding common substructures shared between sequence-structure pairs. Local sequence-structure alignment (LSSA) implements

this, although it is more general and can also allow the exclusion of divergent substructures[13].

1.3.4.4 Simultaneous alignment and folding

Of course, the two above methods can be combined, producing alignments and structures for sequences which have neither. As demonstrated by Sankoff, these problems can be considered rigorously and simultaneously, considering all alignments and folds via dynamic programming [216]. Unfortunately, this approach is $O(N^6)$ even for the two-sequence case, which has meant that most practical approaches have relied on heuristics and approximations. In theory, the steps could be performed sequentially, aligning and then folding or vice versa; however, information about shared structure is important for determining a correct alignment, and covariation in an alignment is a powerful signal in determining a structure. Thus the problems are not truly separable.

Many approaches to solving this type of problem limit the solution space directly, restricting the possible alignments or the types of structures that are evaluated. One of the earliest practical implementations was FOLDALIGN, which used two major restrictions, looking only at non-branching structures (with fully local alignment to look for substructures, since many ncRNAs have branching), and optimizing the number of base pairs, rather than a more complex energy or covariation model [91]. This framework can be used to identify optimal sets of hairpins, corresponding loosely to larger, more complicated structures [127], but later versions relaxed the structural assumptions and incorporated a more sophisticated scoring scheme, while maintaining restrictions on absolute and relative motif length [107]. Dynalign, in contrast, performs global alignment with branching and energy-based folding, but restricts the search based on which positions are allowed to align to each other, essentially creating a fixed-width band around the diagonal of the sequence alignment matrix [176]. STEMLOC featured a very general description of possible constraints, which could use pre-computed limits on the sequence alignment, possible folds for each sequence, or both [121]. CONSAN was a special-case of that approach, using HMM-alignment to constrain a few confidently aligned positions [61].

Alternative approaches to restricting the search space are more indirect; CARNAC parses the sequences into potential stems that are considered as alignment units [197, 242]. Other methods re-frame the problem to provide a simpler calculation, such as operating on pre-computed pairing probabilities for each sequence,

rather than a full folding model, an approach used by PMComp [117] and later LocARNA [258]. A similar approach uses both single-sequence pairing probabilities and pairwise alignment probabilities (Murlet [137], RAF [56]). In either case, these proxy matrices are relatively sparse, providing significant constraint to the overall alignment and folding solution space. Yet more abstract is RNA shapes analysis, which reduces the representation of a structure to its helix topology, and searches for such a topology which is common to the near-optimal shape space of all the sequences, but this does not directly provide or imply an alignment between those sequences [202, 232].

An orthogonal approach to determining both alignment and structure is by iterative profile construction. Profiles for RNAs will be discussed in more detail below (1.3.4.6); for the moment it suffices to think of them merely at the conceptual level, analogous to a statistical sequence profile, but with the additional constraint of a conserved secondary structure. Such a model can be trained and refined via Expectation Maximization over the sequences, iteratively aligning the sequences to the model, and re-estimating the model structure and parameters [70, 94, 96, 213]. A recent implementation of this approach is CMFinder [267].

1.3.4.5 *Ab initio* gene finding

Ab initio prediction of protein-coding genes has been an extensively researched area for many years, taking advantage of known common elements in gene structure such as translation signals, codon patterns, splice site markers, and, when using comparative genome information, conservation patterns between species. It is also possible to make predictions of new non-coding RNA genes as well, although fewer shared sequence elements have been established. Structural RNAs are expected to adopt energetically favorable folds, but thermodynamic stability alone is not sufficient to demark these RNAs in a genome [205]. However, pairwise or multiple genome alignments can provide additional information by examining conserved structure, eliminating many random foldings that could not be distinguished by energy alone (QRNA [206], RNAz [252]). A major limitation of these methods is the accuracy of the underlying sequence alignments, which are constructed with general alignment tools, agnostic of potential secondary structure.

In eukaryotes, a subset of ncRNA genes may be recognized by the differential use of the three major RNA

polymerases (Pol I, II, and III). Although Pol II transcription is the most extensively studied, as it includes all protein-coding genes and many ncRNAs, and Pol I is restricted to an rRNA precursor transcript, Pol III transcribes a large and diverse set of ncRNAs [53]. Known Pol III transcription signals include internal promoter elements (A box, B box, C box), upstream elements (TATA box, proximal sequence element (PSE), distal sequence element), and a poly-T termination signal, although there is some overlapping use of these signals for Pol II transcription as well. There are not currently any gene finders that rely on these elements for *ab initio* prediction of ncRNA genes, but some including the PSE have been useful in screens looking for particular classes of ncRNAs known to be Pol III transcripts [236].

Another approach is less universal, but applies in some organisms with unique features that identify genes in general (either coding or non-coding). If a region has characteristics that identify it as likely to be transcribed and functional, but it cannot be classified as a protein-coding gene via an open reading frame or homology to known proteins, it may be a candidate for a novel non-coding RNA. Examples of where this strategy has been deployed include some thermophiles with AT-rich genomes, where there is more balanced nucleotide composition in genes [141], and the ciliate organism *Oxytricha trifallax*, which in addition to the germ-line genome maintains a second nucleus of many very small chromosomes, each containing generally a single gene [131].

1.3.4.6 Alignment to a sequence with known structure

One of the most common sequence analysis problems is the alignment of new sequences to known members of a family, and its corollary, homology search. In this case, the information about the sequences is asymmetrical: for the known examples of the family, we will typically already have an established structure, but possible new examples will only have primary sequence data. The problem then is to construct alignments that respect both the primary sequence and the given secondary structure.

Family-specific tools One approach to homology search is to have customized search tools for each gene family, explicitly implementing expert knowledge about the conservation of important features of the RNA. This method has been used for many of the largest and most well-studied structural RNAs, such as tRNAs

[167], snoRNAs [168, 217], and microRNAs [152, 163, 269]. This approach is of significant practical importance, particularly in genome annotation, where a specialized tool customized for high sensitivity and specificity to its particular target may be preferred over a more general purpose tool. However, the main strength of these programs is also their greatest drawback: they are highly customized to a particular family, representing a large investment of time and effort to codify the search criteria. Because of this, most smaller RNA families cannot support a specialized tool, and even updating an existing tool with new information about a family can become difficult.

Pattern-matching Pattern-matching tools use specialized description languages to express the consensus structure and sequence of an non-coding RNA [62, 87]. These descriptions are then searched against a database for either exact or inexact matches. Because of the nature of the patterns, which typically place strong limits on the type and number of changes from the consensus, searches with pattern-matching tools tend to be quite fast compared to more general methods. (RNABOB [64], an alternative implementation of the grammar used in RNAMOT [87], remains one of the fastest ncRNA search tools.) Patterns can be written for any RNA, although doing so may require a significant amount of user expertise. Scoring schemes are often yes/no (i.e., is or is not an example of the pattern), which can make it difficult to determine which hits are better than others. RNAMotif [171] does incorporate a method for scoring and ranking sequences on the relative goodness-of-match to the pattern, but the scores themselves are completely user-defined, limiting their utility.

Distance methods A more general approach to the problem is to design a scoring function that describes the cost, or distance, between a reference and a putative new example of the family. The terms describing the distance function generally include features familiar from primary sequence alignment, such as match vs. mismatch and insertion/deletion events, but also include terms describing changes in structure, such as breaking a base pair [128]. Depending on the complexity of the objective function, optimization may be similar to parsimony or weighted parsimony, minimizing the number of events required to describe the history of the sequence, or it may represent a more generalized measure of (dis)similarity. Most distance methods

use dynamic programming algorithms similar to those applied in other folding or alignment problems [44], but other search methods such as genetic algorithms [192] have also been applied. One major difficulty with these methods is that the parameters determining the relative cost of different events have a large effect on the solutions produced, but these numbers are often chosen in an *ad hoc* manner, as there is no theory underlying their selection.

RNA profile models Like profile-HMMs for protein models, there are also probabilistic profile models for RNAs with secondary structure. Conceptually, these models are quite similar and share many important features, including position-specific residue scores, probabilistic insertions and deletions, and known polynomial-time algorithms for alignment. The key difference between the two is that models for RNA include an additional layer of complexity in order to represent the long-range interactions of base-pairing. HMMs belong to a class of models known as regular grammars, which model essentially linear data. The next more complicated set of grammars, called context-free grammars, model nested, branching structures, which were recognized to be a natural fit for the folded structure of non-coding RNAs [70, 213, 214].

Covariance models (CMs) are a specific implementation of a stochastic context-free grammar (SCFG), designed for profile models of structural RNA. Some states in the model produce single residues, corresponding to unpaired positions, but other states can model the two residues of a base pair, capturing information about covarying mutations which can maintain pairing even while the underlying primary sequence changes. CMs will be introduced in more detail in section 1.4.

Like many other parallels between the models, the alignment algorithms for SCFGs are related to their HMM counterparts. The Cocke-Younger-Kasami (CYK) algorithm can produce the maximum likelihood alignment of a new sequence to a CM [123, 133, 268], analogous to the Viterbi algorithm for HMMs. This alignment gives the correspondence of positions between sequences, but also imposes the secondary structure on the aligned sequences (possibly with changes caused by insertions and deletions). The Inside and Outside algorithms correspond to the Forward and Backward algorithms, respectively, and can similarly be used for model training and posterior decoding [63, 155].

Models are built or trained from one or more known examples of a particular family, aligned with a consensus secondary structure. Frequently, the secondary structure is presumed to be known and is supplied by the user, but like with HMMs it is possible to learn the structure or even the alignment via Expectation-Maximization [63, 267].

1.3.4.7 Methods incorporating pseudoknots

CMs, and most of the computational methods so far discussed, only model strictly nested secondary structures and exclude pseudoknots from the structure. This is generally seen as an acceptable loss for the purposes of homology search, because an even more complex class of models (context-sensitive grammars) would be required in order to include pseudoknots, and typically the nested portion of the secondary structure is sufficient for recognition and alignment. Although some programs which consider pseudoknots have been developed [170, 204], the computational requirements are prohibitive for most applications. Any method allowing pseudoknots could be grouped into one of the above categories (for instance, PKNOTS implements thermodynamic folding of single sequences [204], and RNATOPS provides heuristic profile search [124]), but much of the problem space is so far unexplored.

1.4 Covariance models

1.4.1 Covariance models as stochastic context-free grammars

Context-free grammars

Context-free grammars (CFGs) are a class of formal languages, initially studied in work on speech processing and natural language. CFGs have the property of being able to recognize (or produce) strings with multiple related parts, and can also allow these strings to be nested inside each other. One of the simplest examples to understand is the case of parsing sets of matching, nested parentheses. A simple grammar might contain the rules

$$\mathrm{S} \rightarrow (\ \mathrm{S}\ ) \qquad \mathrm{S} \rightarrow [\ \mathrm{S}\ ] \qquad \mathrm{S} \rightarrow x$$

Here, S is a non-terminal – a grammar element which is transformed into symbols by repeated application of rules or productions. By convention, non-terminals are represented by upper-case letters, and since there is only one in this example, it is clear that this is also the 'start' symbol (a more complex grammar might have an explicitly defined start). The right-hand side of each rule indicates what the non-terminal may be transformed into; in this case the first two rules produce a matched set of parentheses or brackets, and regenerate the non-terminal in between. The final rule here generates a only terminal – a variable representing a character that will not be expanded by further productions – represented by a lower-case letter. (The bracket and parantheses symbols in the previous rules are also terminals.)

Starting from the non-terminal S, we can repeatedly apply these rules to produce any sequence of characters that is a valid string in this language.

$$
\begin{array}{lll}
\mathrm{S} & \text{apply } \mathrm{S} \to (\ S\ ) & (\ \mathrm{S}\ ) \\
(\ \mathrm{S}\ ) & \text{apply } \mathrm{S} \to [\ S\ ] & (\ [\ \mathrm{S}\ ]\ ) \\
(\ [\ \mathrm{S}\ ]\ ) & \text{apply } \mathrm{S} \to [\ S\ ] & (\ [\ [\ \mathrm{S}\ ]\ ]\ ) \\
(\ [\ [\ \mathrm{S}\ ]\ ]\ ) & \text{apply } \mathrm{S} \to x; x = \mathrm{a} & (\ [\ [\ \mathrm{a}\ ]\ ]\ )
\end{array}
$$

So, this grammar produces an arbitrarily deep nesting of parentheses and brackets, with a single letter at the center. If we wanted to recognize a string, rather than generate one, there are two ways to go about this which are equivalent in results, but quite different conceptually.

The first method makes use of a 'stack': a memory location where we can add items we wish to store, and we can retrieve the most recently added item. Only the most recent item pushed on to the stack is accessible, but removing it from the stack ('popping' it) reveals the previous item below it. To recognize a string using a stack we use this procedure: read a single character from the string, and determine if it completes any rules, removing items from the stack if necessary and if they match the same rule. If it satisfies the rule, reverse the transformation, taking the left side to be the new current character. Repeat this until no more rules can be matched, and push any current characters onto the stack. At this point, we can read a new character from the string and repeat the procedure.

For the example above, our first character is '('. No rules end with '(', so add it to the stack, and repeat for the next two '['s. The next character is the central 'a', which fits $\mathrm{S} \rightarrow x$, so we replace it with S. No rules end in S, so that is added to the stack as well. The next character, ']', matches the end of $\mathrm{S} \rightarrow [\ S\ ]$, so we check the stack and remove S. That still matches, so we remove the next item down on the stack, '[', which completes the rule; we transform those three characters to S and replace that on the stack. The final ']' and ')' are handled in the same way, and when we reach the end of the string, all that is left is the starting non-terminal S. If there were any characters left over, that could not be matched to the production rules, then the string would be rejected as not matching the grammar. This procedure, although simple, works best with deterministic grammars where there is only one possible interpretation of a string. In a non-deterministic grammar, we may have to repeatedly backtrack and try different possible parses, making this approach inefficient.

The alternative approach reads the entire character string at once, and recursively considers the possible parses of shorter substrings. If we assume that the string does match, i.e. that it can be represented by S, then what rules must we use to generate the observed data? Our string, $x_1..x_7$ has (at x_1 and there's only one rule that includes that, so x_7 must be) (it is) and $x_2..x_6$ must be S. Similarly x_2 is [, so x_6 is] and $x_3..x_5 = \mathrm{S}$. Repeating again leaves us with the conclusion that x_4 must be produced by S, which it can be, validating our initial assumption. Usually, this process is actually carried out in the opposite direction, considering all possible subsequences from shortest to longest, and keeping tables for each non-terminal of which subsequences match it.

The toy grammar I have been using up to this point omits one major feature of CFGs, which is the ability to have elements not only nested but also beside each other, which is necessary for recognizing branching structures. If I change the grammar to be

$$\mathrm{S} \rightarrow \mathrm{S\ S} \qquad \mathrm{S} \rightarrow (\ \mathrm{S}\ ) \qquad \mathrm{S} \rightarrow [\ \mathrm{S}\ ] \qquad \mathrm{S} \rightarrow x$$

then I have a means of generating an unlimited number of occurrences of S, and I can produce much more complex sets of parentheses, which will still be properly nested. For example:

S	apply S → (S) 3 times	(((S)))
(((S)))	apply S → S S twice	(((S S S)))
(((S S S)))	apply S → [S] twice	((([[S]] S S)))
((([[S]] S S)))	apply S → x	((([[a]] S S)))
...	...	...

((([[a]] (((b))) [[cd]])))

Stochastic models

The example grammars in the previous section were qualitative in nature: they described a family of character strings, and any finite-length string could be assigned as belonging to that particular grammar or not. The models we will use for describing RNAs are probabilistic or stochastic models, where each production rule is augmented with a probability for how likely it is. These are true probabilities, with the sum of all possible productions for a given non-terminal equal to one, and the likelihood of a particular string can simply be calculated as the product of the probabilities of the rules used to generate it. (In practice, for reasons of numerical precision, we will more commonly work with log-probabilities.)

The other main difference between the unweighted grammars and the stochastic version is that the latter will generally be designed in such a way that they could produce any possible sequence, albeit with low probability. This allows the calculation of quantitative scoring for all sequences we may encounter, even if they are quite dissimilar from the model.

Covariance models as a special case of SCFGs

Covariance models belong to the class of stochastic context-free grammars (SCFGs), although the terms are not interchangeable. Like profile-HMMs for protein, CMs encode certain assumptions about their domain that aren't inherent in the larger class of models; these range from the obvious, like limiting the characters to the RNA alphabet, to more subtle effects, such as whether moving directly from an insertion to a deletion is permitted.

It should also be noted that CMs are not the only type of SCFG for computational RNA analysis. SCFG-type models can also be used for RNA folding and secondary structure prediction, with either probabilistic or thermodynamic parameters. (With thermodynamic parameters, the model is technically a weighted CFG, rather than a SCFG). Although similar in broad outline, these models may be quite different in detail. Folding models tend to have non-terminals that correspond to structure elements, which may be used many times in a single fold, whereas the profile style of CMs will have non-terminals (or equivalently, states) that correspond to a particular occurrence of a structure, e.g. a specific base pair with sequence positions rather than a general representation of base pairs.

1.4.2 Covariance models as profile models

Covariance models are profile models, and as such they have a set of stereotyped template elements that are used repeatedly in different combinations to produce a model of a particular structure. The set of elements we will use to describe ncRNAs are these:

Description	Typical production
Pair of residues	$\mathrm{P} \rightarrow a\mathrm{W}b$
5' unpaired residue	$\mathrm{L} \rightarrow a\mathrm{W}$
3' unpaired residue	$\mathrm{R} \rightarrow \mathrm{W}a$
Branching in the structure	$\mathrm{B} \rightarrow \mathrm{SS}$
Start of a substructure	$\mathrm{S} \rightarrow \mathrm{W}$
End of a substructure	$\mathrm{E} \rightarrow \epsilon$

with W being a generic non-terminal (i.e. any of the six types of states listed) and ϵ representing the null production (zero residues). These classes of states are sufficient to describe any non-pseudoknotted RNA secondary structure. It is not a minimal set – there are grammars with fewer non-terminals and rules still capable of describing the same structures – but this description is intuitive and corresponds well to the categorizations that humans use when describing RNAs.

Consider how we might make a model that describes the toy alignment at left in figure 1.3. Because it is an ungapped alignment, we can for the moment assume that all columns are consensus positions, and disregard insertions and deletions. By creating a pair-type state for each annotated base-pair, single-stranded states for the unpaired residues, and start, end, and branching states as needed to connect them, we can produce a SCFG-type model to describe the family of sequences, as shown at right in figure 1.3. This structure is what is called a guide tree, a description of the consensus structure that will be expanded to create the full covariance model. Although in this example it was built from an ungapped alignment, for the more general case of gapped alignments the same procedure is applied to the set of consensus columns.

Each state in the guide tree that produces residues has position-specific scoring – the probability that pair state 6 produces an AU pair can be set completely independently of the corresponding probability for the next pair state, and the same applies to the states for unpaired residues. Pair states need not necessarily be canonical pairs; they have complete 4x4 probability matrices, and can describe any observed pattern in occurrences.

So far, this model contains only match states for the consensus positions; a full CM must also handle insertions and deletions for gapped alignments. First, I will consider the case of deletions. A deletion is simply an event where we have no residue in a position where the model expects (at least) one consensus residue. To accomplish this, we will create additional states that can be used instead of the main match state. Because this is an exclusive choice (either there is a residue in the consensus position, or there is not), we will augment each residue-emitting consensus state with an alternative delete state, and group them into a set called a node. The nodes for base pairs present an additional consideration: so far we have a state that produces both residues (match pair) and a state that produces none (delete). However, there is also the case when only one of the two residues has been deleted. This could be modeled as deletion of the pair followed by insertion of the single residue (insertions will be discussed next), but the CM architecture has chosen the other approach, in which the node for a pair also contains two additional states that each emit only one of the residues. The states that we have so far in our nodes (those covering matches and deletions) are also called a split-set, which means that they are exclusive of each other, and we must choose exactly one of them when

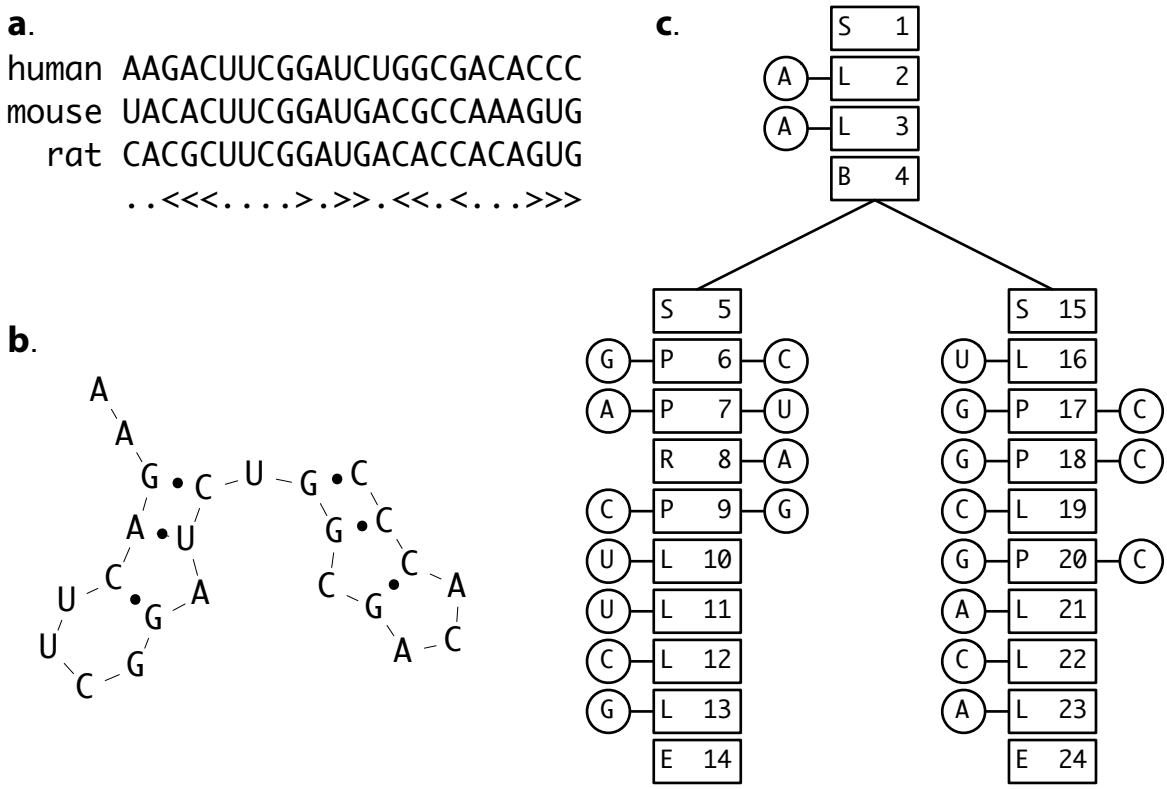


Figure 1.3: **An example RNA family with structure and guide tree. a.** An alignment of three sequences for a toy example of a structural RNA. The final line indicates the conserved secondary structure, with '.' marking unpaired positions and '<' and '>' marking the two halves of a base pair. Because we insist that secondary structures are strictly nested, the structure line is unambiguous – there is only one way to interpret which '<' corresponds to which '>'. **b.** Secondary structure of the 'human' example sequence. **c.** Guide tree for the consensus model. Consensus positions (boxes) are marked with type and a unique number; circles represent the bases (the human sequence is shown), and the linear sequence can be read by tracing around the edge of the tree counter-clockwise starting from the upper left.

aligning a sequence to the model.

All that remains, then, is states for insertions, which are used to produce additional residues between the columns representing consensus positions. Unlike deletions, insertions are used in addition to rather than instead of the match states in the split set. The same insertion state may also be used more than once in a single alignment, to produce an insertion several residues long. By construction, insertions in a CM are unstructured, unpaired residues, and thus are represented by L or R type states. We do, however, need both types: consider adding an internal loop between two pair states. In order to accomplish this, we will need to be able to insert a 5' residue and a 3' residue, and furthermore, we will need to move from the 5' insertion (IL) to the 3' insertion (IR). To reduce ambiguity, not all nodes will have both types of insertion available; match left nodes (MATL) have only IL, and match right nodes (MATR) have only IR [66]. By convention, if IL and IR are both present, there is a path from IL to IR, but not vice versa.

Table 1.1 gives the expansion of the general state types to their corresponding nodes, and a list of specific states for each node. These transformations are stereotyped, and applied in exactly the same way for each example of a state type in the guide tree. Note also that the start (S) type has been divided into three different classes depending on the location of the substructure within the model. This helps to distinguish the global root of the model (ROOT) from other substructures in the model, with BEGL representing the left half of a branching structure, and BEGR the right, and the three classes have slightly varying behavior on which insertions they allow.

As the expansion from a consensus state to a node including match, delete and insert states is stereotyped and predictable, so too are the transitions between these states. Each of the states in the split set of a particular node has a path to any insert states in the node, as well as all of the states in the split set of the next node in the guide tree. Each insert state has a path that returns to itself, and also to each state in the next split set, and as mentioned before IL has a path to IR if both are present in that node. Bifurcation nodes, with their single B state, are special; they transition with probability 1.0 to a pair of explicitly defined BEGL and BEGR states. A view of the fully expanded CM organization is shown in figure 1.4.

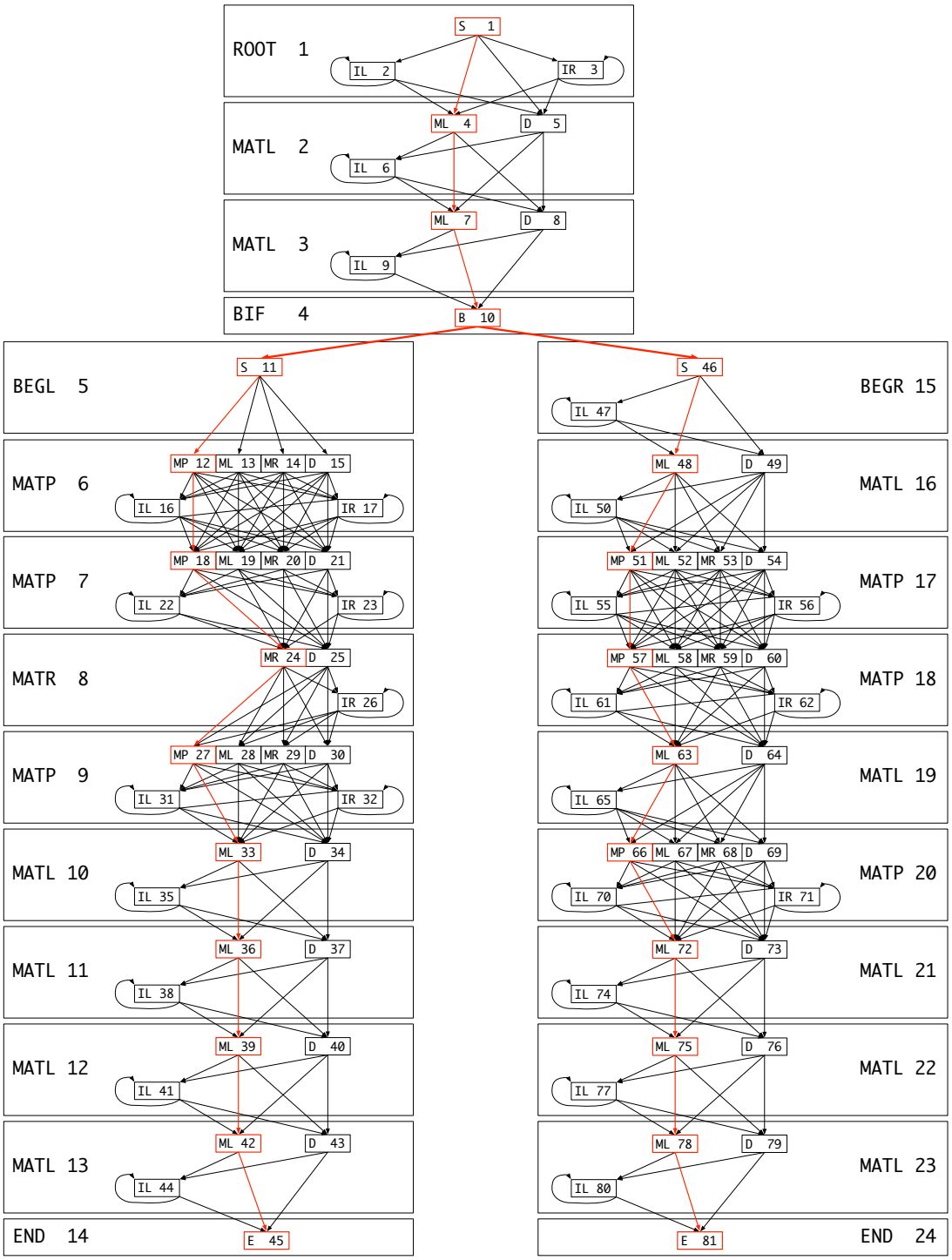


Figure 1.4: **A completely expanded CM.** Larger boxes correspond to nodes, with split set states in each upper row of small boxes, and insertion states in the lower row. The consensus path is highlighted in red.

Table 1.1: **Node types and corresponding states.**

Node	Primary type	States	
		Split set	Insert states
MATP	P	MP ML MR D	IL IR
MATL	L	ML D	IL
MATR	R	MR D	IR
BIF	B	B	
ROOT	S	S	IL IR
BEGL	S	S	
BEGR	S	S	IL
END	E	E	

Given the now complete CM, let us revisit our multiple sequence alignment. Figure 1.5 gives the same alignment as before, with the addition of a more distantly related sequence. At left, the parse for the 'human' sequence is the same as that give in figure 1.3, but the states are now fully labeled with both the node names and numbers (consistent with the numbering of the guide tree), and the individual state numbers which uniquely identify CM states. At right is the parse for the more distantly related 'worm' sequence, requiring the use of various insertion and deletion states, rather than just the consensus states. Although the exact state path varies, both sequences follow the basic framework of the guide tree, conforming to the same overall secondary structure.

1.4.3 Modeling evolutionary changes

Substitutions Substitutions are handled by the position-specific residue scores at each consensus state. These log-odds scores reflect the observed residues in the known sequences, the overall level of conservation, prior information about substitution patterns of RNA, and the background frequency of nucleotides. For single-stranded positions, this score is computed in the same way as match-position scores in HMMs for

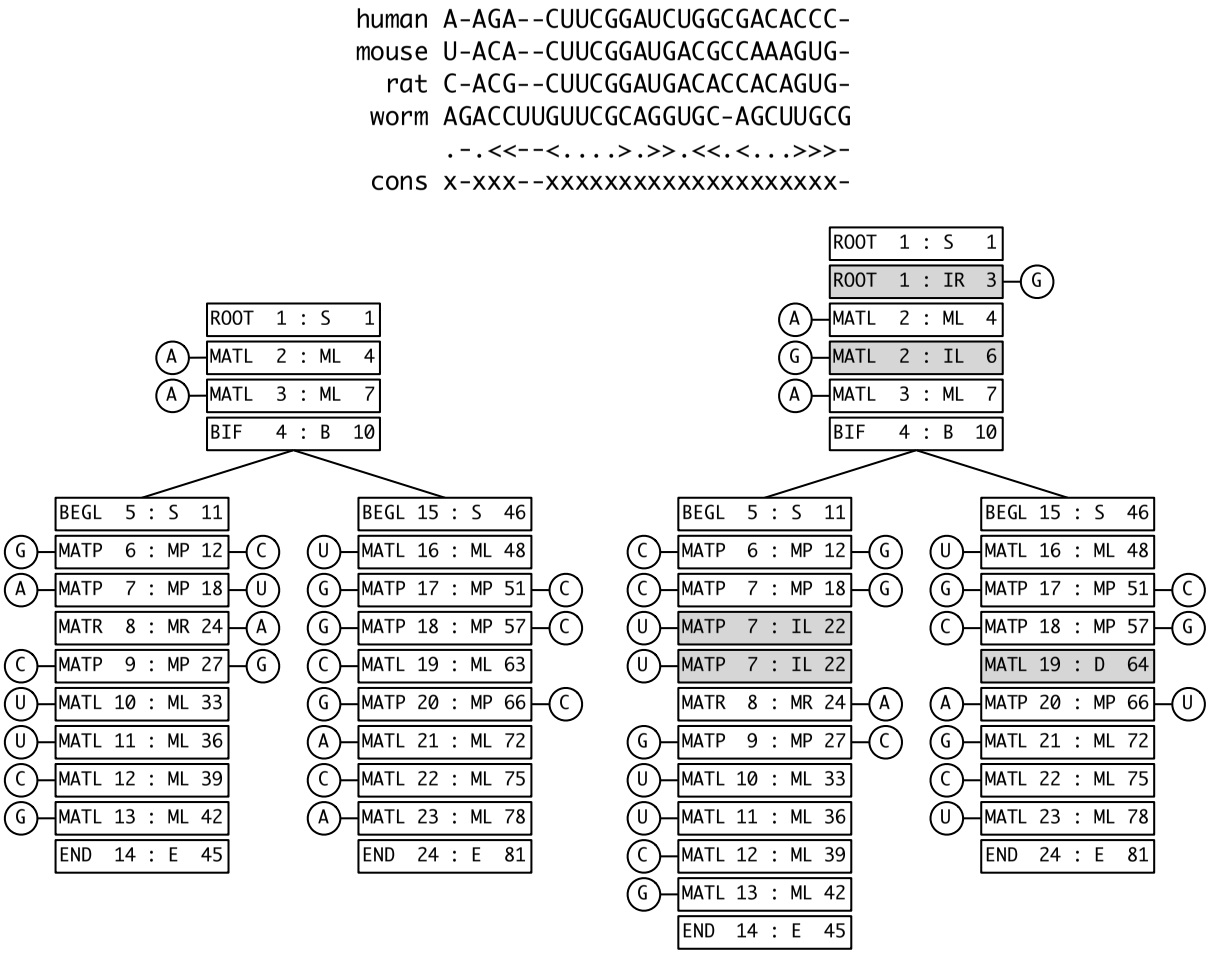


Figure 1.5: **Two sample parses for a complete CM.** Top: a multiple sequence alignment including a more diverged sequence; '-' characters represent insertion or deletion events. The final line, 'cons' indicates which columns were treated as consensus positions in building the model. Left: complete state parse for the human sequence, which uses only consensus states. Right: complete state parse for the 'worm' sequence; non-consensus states are shaded in gray. Each box gives both the node label and number, and the individual state label and number.

protein sequences. Residue scores in pair states, however, reflect the joint probability of both nucleotides occurring together. Frequently, the score matrices for a pair will reflect Watson-Crick base pair preferences, but because the scores are still position-specific and trained on the data, they can also produce scores that favor a highly conserved specific pair, co-varying non-canonical interactions (such as purine-purine pairs), or any other possible combination.

Short insertions and deletions Short indels are handled by the previously described insertion and deletion states. Each consensus position has a corresponding delete state, to bypass that residue, and paths in the model allow moving from delete state to delete state, possibly removing several residues in a row before rejoining the main consensus match states. Insertions are handled by special states between match positions; these states emit a single residue, but have a circular path to return to themselves to produce additional residues. The term 'short' for the indels produced by these states is relative – there is no particular limit on their lengths, however there is a crossover point (which varies per model and per position) at which it is beneficial to the score to use the alternative handling for longer events, discussed next.

Long insertions and deletions When the changes in a sequence relative to the consensus are large, and involve the addition or removal of entire substructures, a different method for handling insertions and deletions, called local end and local begin. These two elements are a core part of local alignment with CMs, rather than global or glocal [1] alignment. Local begin allows the alignment to skip from the start (root) state of the model, to any consensus state anywhere in the structure. This allows the alignment to exclude the 'outer' portions of the sequence (5' and 3' ends), and align only to a smaller substructure. Local end is similar, but it allows the alignment to move from any consensus state to an alternative end state. This removes the rest of the substructure, and optionally replaces it with a length of unaligned sequence. Parses showing the use of local begin and local end are shown in figure 1.6.

[1] Glocal, or global-local alignment, is global with respect to the model but local with respect to the sequence; it allows for the case when a complete match to the model is embedded in a longer region of non-homologous sequence.

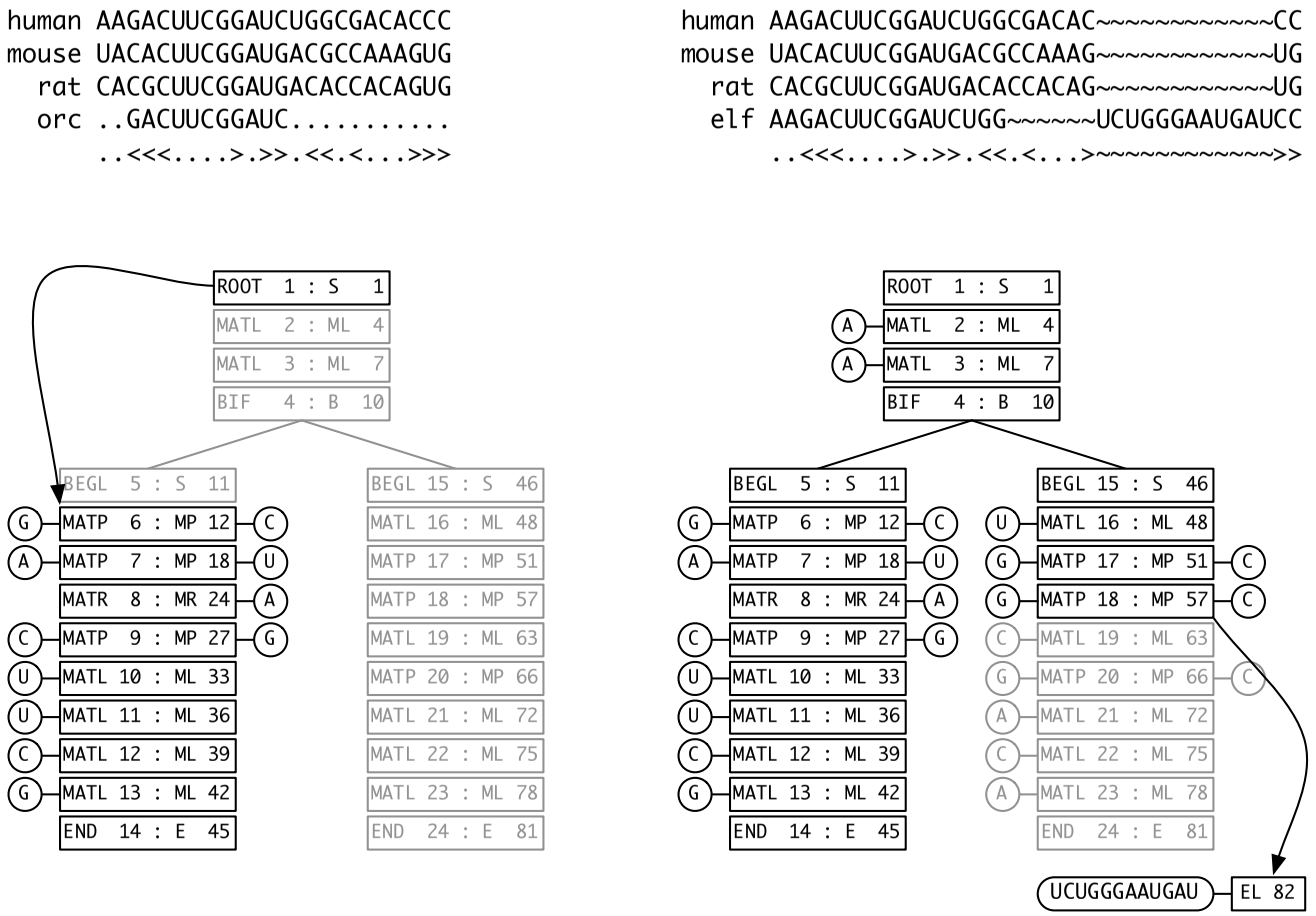


Figure 1.6: **CM parses using local begin and local end. Left:** an alignment using local begin to match a substructure of the model. **Right:** an alignment using local end to replace a substructure. The 'elf' sequence here has a substantially longer stem-loop than the reference examples, and it does not match well in sequence, so the alignment skips a subsection of the model, replacing it with a long, unstructured insertion. In the alignment, '∼' indicates that the positions are not aligned, and is used to visually distinguish between single-position indels and this longer local-end-type indel.

1.4.4 Building a CM

Input

The minimal information required in order to build a covariance model is a single sequence with an associated secondary structure [140]; commonly, though, it will be a multiple sequence alignment with a consensus structure. The structure is given as an additional line of the sequence alignment, with paired brackets indicating paired elements. (The exact details of data formatting are not particularly relevant for this overview; specifications are given in the Infernal User's Guide [188].)

Consensus positions

The first step of model building is to determine which of the alignment columns correspond to consensus positions that are generally expected in members of the sequence family. By default in Infernal, the possible consensus residues are evaluated under a simple heuristic: columns in which at least 50% of the sequences have a residue, rather than a gap, are taken to be the consensus, and the remainder are taken to be insertions. It is also possible to manually specify which columns represent the consensus. (In the case of a single sequence, all positions are treated as consensus.)

Model priors

The model priors are all the parameterization information that didn't come from the input sequence(s): knowledge about 'typical' RNAs, such as pairing preferences, and relative frequencies of insertions and deletions. For single-sequence models, the RIBOSUM matrices represent a general model of RNA evolution, similar to scoring matrices for proteins [140]. The priors for multi-sequence models were derived in a similar fashion, based on observed changes in trusted RNA alignments [185]. Mixture Dirichlet models are used for the priors, so that there are multiple classes of positions learned from typical RNAs, and the observed data can then be fit to the most appropriate class.

Parameterization

Parameterization is the process of combining the general information from the prior with the specific information from the sequences. Each consensus model position is assigned probabilities for the possible residues it can produce based on a mixture of the prior and the observed frequencies in the alignment using posterior mean estimation (in the absence of a prior, this would simplify to a maximum likelihood estimate based on the observed data). The relative weights for the prior and the observations can be varied, although a reasonable default is provided; more weight toward observation produces a model that is more specific for sequences very similar to the ones it has seen, more weight toward the prior gives a 'fuzzier' model that can be more sensitive to sequences more divergent than any that were in the training set. Transition probabilities between states are adjusted to reflect whether this family is more or less willing to accept insertions and deletions at each position.

Calibration

Calibration is technically both optional and separate from model creation, but it is an important step in preparing the model for practical use. The calibration step runs simulations of scores for alignments of random sequences, in order to determine the parameters that describe the score distribution of this particular model. Knowing these parameters enables searches to calculate P-values and E-values, rather than relying on the raw score of the alignment. This step is also required in order to use many of the filtering features that increase the speed of CM searches.

1.4.5 Using CMs in sequence analysis

Alignment

Alignment to an SCFG is conducted by the previously mentioned CYK algorithm; the form used for CMs is as given below. The notation follows that of Eddy 2002 [66], with $\alpha_v(i, j)$ being the alignment score of model state v to subsequence (i, j); $\mathcal{S}_v$ being the type of state v, and $\mathcal{C}_v$ being the set of child states from v –

those states which v has a non-zero transition probability to. The transition and emission probabilities are t_v and e_v, respectively.

for $v = \mathrm{M}$ **down to** $1,\ j = 1$ **to** $\mathrm{L},\ d = 1$ **to** j

$i = j - d + 1$

if $\mathcal{S}_v = \mathrm{D}$ or S

$\alpha_v(i,j) \ = \ \max_{y \in \mathcal{C}_v} [\alpha_y(i,j) + \log t_v(y)]$

else if $\mathcal{S}_v = \mathrm{P}$ **and** $d \geq 2$

$\alpha_v(i,j) \ = \ \log e_v(x_i, x_j) + \ \max_{y \in \mathcal{C}_v} [\alpha_y(i+1, j-1) + \log t_v(y)]$

else if $\mathcal{S}_v = \mathrm{L}$ **and** $d \geq 1$

$\alpha_v(i,j) = \log e_v(x_i) + \max_{y \in \mathcal{C}_v} [\alpha_y(i+1, j) + \log t_v(y)]$

else if $\mathcal{S}_v = \mathrm{R}$ **and** $d \geq 1$

$\alpha_v(i,j) = \log e_v(x_j) + \max_{y \in \mathcal{C}_v} [\alpha_y(i, j-1) + \log t_v(y)]$

else if $\mathcal{S}_v = \mathrm{B}$

$(y,z) = \mathcal{C}_v$

$\alpha_v(i,j) \ = \ \max_{i-1 \leq k \leq j} [\alpha_y(i,k) + \alpha_z(k+1, j)]$

else if $\mathcal{S}_v = \mathrm{E}$ **and** $d = 0$

$\alpha_v(i,j) \ = \ 0.0$ *(Initialization)*

else

$\alpha_v(i,j) \ = \ -\infty$ *(Initialization)*

For global alignment, the score of the best alignment is $\alpha_1(1, L)$ and the alignment itself may be traced back as the (v, i, j) triples which produced that score. The various forms of local alignment have slightly different initialization and termination conditions, but the variations are generally minor.

Alignment for CMs is performed by the `cmalign` program, which takes arguments of a CM and a sequence file containing one or more new sequences to be aligned, and gives output as a multiple alignment. `cmalign` has additional options for alignment algorithms, such as optimal accuracy or posterior sampling

[63, 122, 188], but this work will generally take alignment to refer to CYK.

Search

Alignment is most appropriate when a set of likely homologs have already been identified; for the purposes of database search, a 'scanning' variant of CYK is typically used. The algorithm is basically the same, with two key differences: the length of alignments d is limited to some maximum window, and the order of loops is changed so that j rather than v is outermost. This changes the nature and amount of information that needs to be stored when dealing with large sequences, but otherwise does not alter the scores of alignments.

The implementation of search for CMs is, predictably, `cmsearch`; it can be used with or without a variety of filtering strategies to reduce the effective database size. These methods will be described in more detail in chapter 5.

Interpreting scores

Two types of scores are available for CMs, called CYK and Inside scores for the algorithms used to calculate them. CYK scores give the likelihood that a particular sequence and its best alignment were produced by the model. The Inside score, by contrast, gives the likelihood that a particular sequence was produced by the model, considering all possible alignments. The Inside score is more powerful, particularly for detecting remote homologies, but it is slower to calculate. Both types of scores are typically expressed as log-odds scores, measuring the relative likelihood between the model of homologous sequences (the CM) and the model of non-homologous sequences (the background or random model). These scores can be given as raw values (bitscores, usually), or they may be converted to significance estimates in the form of E-values.

Rfam

In addition to any arbitrary model which may be created, the Rfam database is a collection of known structural RNA families, for which multiple sequence alignments with consensus structures and pre-built CMs are provided [85, 99]. Rfam is constructed around the concept of seed and full alignments, where seed

alignments feature a relatively small number of example sequences which sample the observed diversity in known sequences. Curation is performed exclusively on the smaller seed alignment, and the full alignment is constructed automatically by search of a sequence database with the seed model. Internal quality control measures are used to ensure that the seed models find all sequences which have previously been identified, but if new sequences are identified experimentally which the current model misses, the seed can be updated relatively easily so that it will include the new examples. The most recent release of Rfam (10.0, April 2010) includes 1446 models and is built with Infernal 1.0, but when Rfam is used later in the work, that will refer to the earlier 9.1 release (1371 families, December 2008).

1.4.6 Limitations of CMs

As powerful as CMs have proved to be for sequence analysis of structural RNAs, they do have some caveats, the most noteworthy of which are given below.

Lack of pseudoknots

CMs only deal with completely nested structures, and cannot handle pseudoknots. This limitation is common among many of the computational analysis tools for structural RNA, but it is important to be aware of as many important RNA interactions involved these non-nested structures. Usually, in annotating the consensus structure for an RNA family, the curator will attempt to find the largest subset of the known base pairs which can be represented as completely nested. The remainder of the base pairs can be annotated for humans looking at the structural alignment, and CMs will maintain these annotations, but the residues will be aligned and scored as single positions rather than pairs. Because of the vastly higher complexity of including proper models of pseudoknots, this trade-off is generally seen as acceptable for search and recognition of homologs, and okay for alignment (although a human editor may wish to refine the CM alignment, with a particular eye towards the pseudoknotted interactions).

Ignoring phylogeny

CMs share an important limitation with HMMs, which is the independence assumption on the sequences which make up the training alignment. The training procedures fit a model in which each example sequence is a completely independent sample from a single distribution, equivalent to the assumption of a star phylogeny. This ignores the real and non-uniform relationships in the true evolutionary history of the sequences. Current training procedures address this problem by means of sequence weighting, which reduces the relative contribution of sequences which are very similar to each other. This reduces the magnitude of the issue, but does not address the underlying assumption, and new sequences aligned to the model are also assumed to be independent samples, rather than potentially being grouped into an appropriate clade. Some work has been done on combining profile-based and phylogenetic models for primary sequence but this is still an active area of research with uncertain results [22, 200].

Large insertions

It is perhaps surprising to give large insertions as an example of the limitations of CMs, when I have previously described the methods used to accommodate them (see 1.4.3). However, some less-than-desirable behaviors arise from the current handling of large structural changes, particularly in the case of large insertions relative to the model. First among these is that insertions are modeled as unstructured primary sequence, with a single nucleotide frequency for all residues, even at lengths of 100 or more bases. Clearly, RNA of that length is unlikely to exist *in vivo* as a completely unfolded loop, but without relationship to the consensus columns of known structure, a CM has no method of assigning additional base pairs.

A corollary to this is that because every residue in a long insertion is modeled by the same single state, there is no clear correspondence between insertions in different sequences, even if they occur at the same position in the model. Because each sequence is aligned to the model, and not the other individual sequences, the strongest claim that the model provides is that two subsequences map to the same insertion state, without any finer-resolution mapping of the relationship of the residues within in those insertions.

Computational time

The limitations described above are largely conceptual ones – they reflect potentially problematic shortcomings in the design and structure of the models. The final and perhaps most pressing point that I will mention is the practical limitations imposed by the resource requirements of actually running the search and alignment algorithms. Because they have exact polynomial-time solutions, CM alignment and search are 'efficient' in the technical sense, in that they do not need to use brute force examination of all solutions, and produce a globally optimal solution. However this does not imply that running them is necessarily feasible. The CYK algorithm given previously has computational time requirements that scale as $O(N^4)$: if one structural RNA alignment is approximately twice as long as another, aligning new sequences of that length will take on the order of 16 times as long. This scaling behavior can make even moderately sized models intractable without considerable computational resources. These time requirements make CM-based search and alignment unattractive for direct application in many cases, and they are often preceded by sequence-based filtering approaches to reduce the amount of the database that is subject to the more intensive search technique, despite trade-offs in sensitivity.

1.5 Evaluating the significance of alignments

1.5.1 Distinguishing between real and random alignments

All the various alignment and comparison methods listed above produce a score (probabilistic or otherwise) quantifying the similarity between the sequences. Usually, we will want to interpret that score to make some judgement about the relationship; for example, is the score high enough to infer homology? The scale of the problem is magnified in the case of database searches, where every sequence will be assigned a score, and we will want to use those scores to determine a group of those sequences for which the level of similarity is meaningful or relevant, but the sheer number of comparison means we are more likely to find a high-scoring alignment by chance.

In a fully probabilistic framework, where we are explicitly considering the probability of an alignment

under a model of similar sequences ($P(x, y|M)$) and under a model of independent random sequences ($P(x|R)P(y|R)$), we can take a Bayesian approach. Under Bayes' rule, if we have prior probabilities on the likelihood of homologous sequences ($P(M)$) versus the likelihood of random sequences ($P(R) = 1 - P(M)$), the probability that the homology model is correct for a particular alignment is

$$P(M|x, y) = \frac{P(x, y|M)P(M)}{P(x, y)} = \frac{P(x, y|M)P(M)}{P(x, y|M)P(M) + P(x, y|R)P(R)}$$

The prior probabilities might vary depending on the context of the search – for instance whether we expect to find a single hit, or ten, or 1% of all sequences.

The alternative approach to estimating the significance of a score is a classical statistics framework, which is a two-step process. First, a particular scoring model is characterized to determine its cumulative score distribution $F(S)$, the fraction of scores that are less than S. Then, given a score and the number of independent comparisons that have been made, we can directly calculate the number of times we expect to see a score at least this large. If this expectation number $E(S)$ is much less than 1, we can consider the score as unlikely to have been produced by a random sequence. This approach is applicable to both *ad hoc* and probabilistic scoring systems, and it is the approach I will use throughout this work when needing to determine the significance of scores.

1.5.2 Extreme Value Distributions

Characterizing the cumulative score distribution function typically means fitting the parameters of a standard statistical distribution to the observed data. This does imply that the score distributions do in fact conform to a recognized type of distribution; if the functions were arbitrary, the approach would be impractical. Very early work treated these scores as Normally distributed, but it quickly became apparent that there were more high scores in alignments of unrelated sequences than would be expected, and that this distribution over-estimated the significance of scores [225].

In practice, maximum scores for local alignments conform to a Type I Extreme Value Distribution (EVD), also known as a Gumbel distribution. The Gumbel distribution is known to describe the maximum value

observed in a sample taken from several different statistical distributions, including Normal and Poisson. The Gumbel is similar in overall shape to the Normal, but it has a slower decay of density in the right-hand tail. The probability density function, cumulative distribution function, and survivor function (inverse CDF) for the standard forms of Normal and Gumbel distributions are given in figure 1.7.

The proof that maximal scores for ungapped local sequence alignment follow an EVD was developed by Karlin and Altschul, and given more formally by Dembo and Karlin [51, 132]. Although it has been been observed to hold for certain ranges of gap scores, it has never been proven that the scores of gapped alignments should follow a Gumbel distribution. (Note that this discussion pertains specifically to maximum likelihood local alignment scores, not total likelihood scores like HMM Forward scores or CM Inside scores, which do not fit an EVD.)

1.5.3 Karlin-Altschul theory

A large body of work on the statistics of primary sequence alignment originates with Karlin and Altschul in the early 1990s, beginning with the result mentioned above for the EVD approximation for the distribution of maximal scores of ungapped local alignments. The probability of encountering a score S greater than x is usually given in the form

$$P(S \geq x) \approx 1 - \exp\left(-Kmne^{-\lambda x}\right)$$

This is simply a modified form of the general expression of an EVD

$$P(S \geq x) = 1 - \exp\left[-e^{-\lambda(x-\mu)}\right]$$

but the coefficients can be assigned particular meaning in the context of sequence alignment. For ungapped alignment, λ is the unique positive solution to the equation

$$\sum_{a,b} f_a f_b \exp\left(\lambda s_{ab}\right) = 1$$

with f_a, f_b being the background frequencies of residues a, b, and s_{ab} the score of aligning a to b, and the interpretation that λ is the natural log base of the scoring system. λ describes the rate of decay in observed

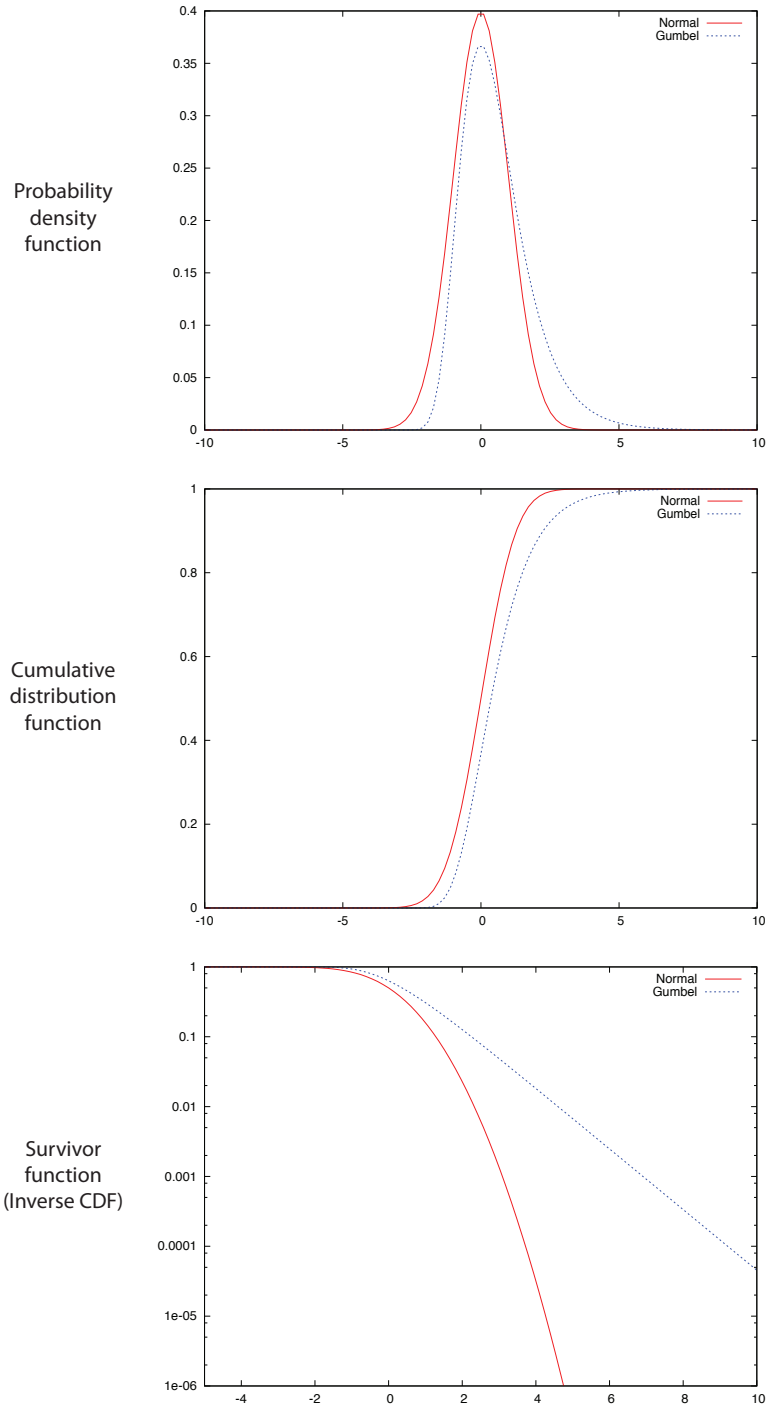


Figure 1.7: **Normal and Gumbel distributions.**

maximal scores; i.e., it is the slope parameter of the EVD and controls the variance of the distribution. The second parameter μ describes the location of the distribution, being equal to the mode. In this case

$$\mu = \frac{\ln(Kmn)}{\lambda}$$

with m and n being the lengths of the two sequences (their product corresponding to the number of start positions in the alignment matrix), and K describes the relative independence of these different starting locations. For ungapped sequence alignment, K may also be calculated directly from the scoring matrix and base frequencies, but in the case of gapped alignment it and λ must both be found by simulation [5, 132].

Frequently, the score S is transformed by $S' = \lambda S - \ln(Kmn)$ which reduces the the expression of the EVD to its standard form with mode 0 and scale 1:

$$P(S' \geq x) = 1 - \exp\left[-e^{-x}\right]$$

For $x > 2$ (which corresponds to approximately 2.5 standard deviations), this can be closely approximated by $P(S' \geq x) \simeq e^{-x}$.

Further refinements of this theory included the calculation of edge effects. The description above assumed that all mn cells represented possible starting points of the alignment, however high-scoring alignments do not exist as a point but have an associated length. Thus it is difficult or impossible for cells very near the edges of the matrix to produce the optimal alignment simply because there does not remain enough sequence to form a sufficiently high score. This introduces the concept of effective sequence lengths m' and n', described as

$$m' \approx m - \frac{\ln(Kmn)}{H}$$

where H is the entropy of the scoring system, meaning $\frac{\ln(Kmn))}{H}$ is the expected length of a high-scoring pair [5].

Chapter 2

trCYK: Local RNA structure alignment with incomplete sequence [1]

2.1 Abstract

Motivation:

Accuracy of automated structural RNA alignment is improved by using models that consider not only primary sequence but also secondary structure information. However, current RNA structural alignment approaches tend to perform poorly on incomplete sequence fragments, such as single reads from metagenomic environmental surveys, because nucleotides that are expected to be base paired are missing.

[1] This chapter was originally published as D. L. Kolbe and S. R. Eddy. Local RNA structure alignment with incomplete sequence. *Bioinformatics*, 25:1236–1243, 2009; the previously published portion is reproduced without modification. The supplemental material in section 2.8 was not included in the original publication, and is authored solely by DLK.

Results:

We present a local RNA structural alignment algorithm, trCYK, for aligning and scoring incomplete sequences under a model using primary sequence conservation and secondary structure information when possible. The trCYK algorithm improves alignment accuracy and coverage of sequence fragments of structural RNAs in simulated metagenomic shotgun datasets.

Availability:

The source code for Infernal 1.0, which includes trCYK, is available at `http://infernal.janelia.org`

2.2 Introduction

Sequence alignment approaches may be broadly divided into *global alignment* methods, where sequences are assumed to be homologous and alignable over their entire lengths, and *local alignment* methods, where only part of each sequence is assumed to be homologous and alignable [63, 103]. Local alignment is more widely used because there are many biological and technical reasons why sequences may not be globally alignable. For example, many protein sequences have arisen by accretion of common protein domains in different combinations [247], and some high-throughput sequencing strategies such as metagenomic shotgun sampling generate fragmentary sequence data [218].

For local alignment of primary sequences [6, 196, 224], one is merely looking for an alignment of contiguous linear subsequences of a query and a target. At this level there is little informative difference between local alignments that arise by biological evolution versus incomplete data. However, the nature of local alignment can be markedly different when we adopt more realistic and complex sequence alignment models that capture evolutionary constraints at a higher level than primary sequence alone. For example, in comparing three-dimensional protein structures, which often share structural similarity in only part of their overall fold [41], it is advantageous to adopt local structural alignment algorithms that allow alignment of spatially local units of three-dimensional structure that may not be composed of contiguous colinear residues in the primary

sequence [88].

Here we are concerned with alignment of structural RNAs, using models that consider both primary sequence and secondary structure constraints. Evolution of a structural RNA is constrained by its secondary structure. Base pairing tends to be conserved even as the sequence changes, and aligned sequences exhibit correlated substitutions in which base pairs are substituted by compensatory base pairs. Computational methods for aligning structural RNAs under a combined primary sequence and secondary structure scoring model have been developed [13, 213] including "covariance model" (profile stochastic context-free grammar) methods [63, 66, 70, 185]. These models represent a given RNA consensus secondary structure as a binary tree, with individual nodes representing and scoring individual base pairs and single-stranded residues.

Local RNA secondary structural alignment has been implemented by allowing an alignment to start or end at any internal node in the tree [13, 66, 140], much as local primary sequence alignment allows starting and ending at any residue in the linear sequence. This *subtree method* of local RNA alignment can include or exclude any subtree of the RNA, corresponding well to secondary structure domains. Biologically, this serves as a reasonable approximation of some important evolutionary constraints on RNA secondary structure alignment, although it neglects higher-order constraints, including pseudoknots and tertiary structure.

However, defining locality by subtrees is a poor model of local structural RNA alignment when locality arises for technical rather than biological causes. A shotgun sequencing strategy will truncate at the linear sequence level without respect for the conserved base-paired structure, and residues involved in base pairs may be missing in the observed sequence, as illustrated in Figure 2.1. In this case, we do not want to score the missing residues as deletions of conserved base pairs, but neither do we want to leave the homologous observed residues unaligned if we are trying to get the most information from fragmentary sequence data.

Here we will focus specifically on the *homology search and alignment* problem. We have a given RNA sequence and secondary structure as a query, and the task is to search a sequence database for homologous sequences and/or align target sequences to the query. This is directly analogous to the use of the Smith/Waterman local alignment algorithm for primary sequence analysis [224], and it is the problem addressed by our Infernal software package [187], for instance using Rfam models of known RNA families to

infer and annotate homologous RNAs in genome sequence [85]. It should not be confused, for example, with the related problem of *de novo motif identification*, which arises in RNA analysis when the input data consist of two or more sequences that are presumed to share an *unknown* structural motif in common, and the task is to produce a local structural alignment that identifies the common motif and infers its common structure. De novo motif identification requires a means of inferring the unknown structural consensus in addition to a local alignment algorithm. Although we expect that *de novo* motif identification approaches such as CMFinder [267] or other approaches for inferring locally conserved RNA structure such as LocARNA [258] would be able to incorporate the local alignment algorithm we will describe, for the purposes of introducing our local alignment algorithm in the present paper, we will not discuss the *de novo* motif identification problem further.

An important example of the local RNA alignment problem in homology search and alignment arises in metagenomic shotgun survey sequencing [38, 218], particularly when assembly is incomplete or not possible. Structural RNA sequence alignments (particularly of small subunit ribosomal RNA) are important in analyzing the phylogenetic diversity of metagenomic samples, but a single shotgun read (often of only about 200-400 bp) will fall more or less randomly into the consensus alignment of an RNA, generally leaving unsatisfied consensus base pairs because of the incomplete nature of the sampled sequences, and it may also include extraneous genomic sequence.

To deal with this *truncated sequence* type of locality we want to align the observed sequence, or a subsequence of it, to a contiguous subsequence of the yield of the model's tree: the linear consensus sequence, as read counterclockwise around the tree's leaves. We want to use secondary structure information wherever we have both residues in a base pair, but revert to primary sequence alignment when we are missing sequence data. If we magically knew *a priori* the endpoints of the correct alignment of an observed sequence read with respect to the yield of the RNA model, we could derive a new model that used base pair states where we had both residues, and converted pair states to appropriately marginalized single-residue states where the pairing partner was missing. The problem is that these endpoints must be inferred when we align the observed sequence to the model. We describe an optimal recursive dynamic programming solution for this problem, and evaluate the algorithm's utility in accurate alignment of simulated datasets of unassembled metagenomic

sequence.

2.3 Approach

2.3.1 Local alignment as a missing data problem

We frame the alignment of truncated sequences as a missing data inference problem [210]. We specify two probabilistic processes: one that generates complete data (our existing probabilistic model of global alignment), and one that generates observed fragments from the complete data (by random sequence truncation). The joint probability of observed sequence fragments and their local alignment to the model will then be an appropriate marginal sum over global alignments. We will identify the optimal local alignment for the observed sequence by maximizing this joint probability.

We will describe the essence of the approach (and two approximations we make) in general terms with respect to binary trees, deferring the specific notation we use for profile stochastic context-free grammars (covariance models, CMs). In a CM, both the consensus structure of the model and a particular structural alignment of the model to an individual RNA sequence are binary trees. (A binary tree suffices to capture all nested base-pairing correlations, but non-nested interactions such as pseudoknots and higher order interactions such as base triples are neglected [63].) Construction of a CM starts by representing the RNA consensus structure as a **guide tree**, with **nodes** representing consensus base pairs and consensus unpaired positions. Each consensus node is then stereotypically expanded into one or more stochastic context-free grammar (SCFG) **states**, with one state representing the consensus ("match") behavior and additional states and state transitions representing the probability of insertions and deletions relative to consensus. A CM is a special case of SCFGs, with all its states and state transitions arranged in a directed graph following the branching pattern of a consensus RNA structure's binary tree. An alignment of the CM to a particular sequence is represented as an SCFG *parse tree*, a state path through the consensus guide tree, using match, insert, and delete states to account for alignment positions, and start, bifurcation, and end states to account for the branching tree structure itself.

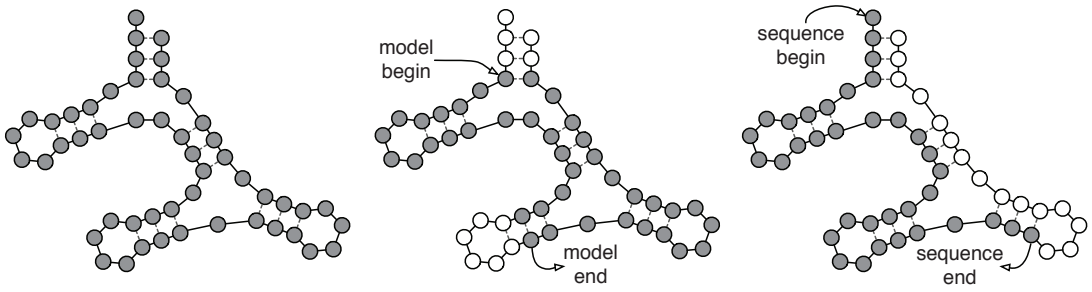


Figure 2.1: **Comparison of local alignment types. Left:** global alignment; filled circles indicate observed residues in an RNA structure, which can be thought of as a binary tree. **Center:** subtree method of local RNA structural alignment. Whole domains of the RNA structure may be skipped (open circles indicate consensus positions without aligned sequence residues), but the observed alignment satisfies all expected structural constraints: if a residue is aligned to a pair state, another residue will be aligned to form a base pair. **Right:** truncated sequence method of local RNA structural alignment, where the observed sequence may begin and end anywhere with respect to the consensus RNA structure. Aligned residues may be base-paired to positions that are missing from the alignment.

A parameterized RNA CM θ specifies a joint probability distribution $P(\hat{\mathbf{x}}, \hat{\pi} \mid \theta)$ over *complete* sequences $\hat{\mathbf{x}}$ and parse trees $\hat{\pi}$: i.e. over *global* alignments.

A missing data process $P(\mathbf{x}, \pi \mid \hat{\mathbf{x}}, \hat{\pi}, \theta)$ specifies how a complete sequence $\hat{\mathbf{x}}$ with length $\hat{\mathbf{L}}$ is truncated to an observed sequence fragment $\mathbf{x}$ of length $\mathbf{L}$, and correspondingly, how the global parse tree $\hat{\pi}$ is truncated to a notion of a local parse tree π. (We will solidify our definition of a local parse tree shortly.) Because we are imagining a complete sequence randomly truncated to a sequence fragment, the missing data process would ideally be conditionally independent of the model and the parse tree. For instance, we could sample each possible sequence fragment from a complete sequence of length $\hat{L}$ with uniform probability $\frac{2}{\hat{L}(\hat{L}+1)}$. However, under this missing data process, we would need to marginalize (sum over) all possible complete sequences of all possible lengths $\hat{L} = 0 \ldots \infty$. This $\frac{2}{\hat{L}(\hat{L}+1)}$ term becomes problematic in the recursive dynamic programming optimization framework we describe below.

Instead we will make what should be a reasonable approximation of the truncation process. We assume that a truncation Δ^{gh} is done by selecting a fragment $g \ldots h$ relative to the positions in the fixed-length *consensus yield* as defined by the *model* (the consensus sequence positions defined by the CM's consensus guide tree nodes). This truncation is then conditionally independent of both the parse tree and the sequence. This approximation should be reasonable because high-probability complete sequences $\hat{x}$ will generally have lengths similar to the consensus length. It means that local alignments will only begin and end at consensus positions, never at sequence insertions. For a model with W consensus positions, the probability of choosing any particular fragment $g \ldots h$ with respect to the complete yield $1..W$ is $P(\Delta^{gh} \mid \theta) = \frac{2}{W(W+1)}$. This term is now a constant with respect to the necessary summation over complete data.

Now we define what we mean by a local parse tree fragment π. Choose two positions g, h on the consensus yield of the model: these consensus sequence positions correspond unambiguously to states s_g and s_h in parse trees (the states used by the parse tree to account for how the endpoints of a particular sequence align to a model consensus position: either a consensus match, or a deletion). A **local parse tree** π^{gh} (equivalent to what we have called just π until now) is defined as the minimal (smallest) subtree of a complete parse tree $\hat{\pi}$ that contains s_g and s_h. Usually this is a parse subtree rooted at either s_g or s_h, but s_g and s_h may also be

on opposing sides of a bifurcation, with the minimal subtree rooted at the bifurcation state.

Truncation of a complete parse tree $\hat{\pi}$ to a local parse tree π^{gh} defines two different sorts of missing data. Outside the local parse tree, we are missing (and will sum over) both sequence residues and parse tree states that were in the complete parse tree; let this missing data be represented by $\mathbf{x}', \pi'$. Inside the local parse tree, we may have states with unsatisfied, missing sequence residues, such as base pairs where only one residue is in the observed sequence: here we will be summing only over the missing sequence residues, denoted as $\mathbf{x}''$. The combination of the observed data $(\mathbf{x}, \pi^{gh})$ and the unobserved data $(\mathbf{x}'', \mathbf{x}', \pi')$ together uniquely determine the complete alignment $\hat{\mathbf{x}}, \hat{\pi}$, with the the truncation Δ^{gh} conditionally independent of both $\hat{\pi}$ and $\hat{\mathbf{x}}$.

The desired joint probability may then be written as a summation over the two types of missing data defined by a local parse tree:

$$P(\mathbf{x}, \pi^{gh} \mid \theta) = \sum_{\mathbf{x}''} \sum_{\mathbf{x}', \pi'} P(\Delta^{gh} | \hat{\mathbf{x}}, \hat{\pi}, \theta) P(\hat{\mathbf{x}}, \hat{\pi} | \theta).$$

Summation over missing data $\mathbf{x}', \pi'$ results in two terms. The first is a term $P(\pi_{x_1} = s_g, \pi_{x_L} = s_h \mid \theta)$ that represents the marginal probability that a complete parse tree truncated at g, h has states s_g, s_h assigned to the endpoints of the truncation; this is just the fraction of complete parse trees that contain states s_g and s_h. The second term is $P(\mathbf{x}, \mathbf{x}'', \pi^{gh} | \pi_{x_1} = s_g, \pi_{x_L} = s_h, \theta)$ for the local parse tree and its associated sequence residues (both observed and unobserved) conditional on local parse tree endpoints at states s_g, s_h. Thus:

$$\begin{aligned} P(\mathbf{x}, \pi^{gh} \mid \theta) &= \frac{2}{W(W+1)} P(\pi_{x_1} = s_g, \pi_{x_L} = s_h \mid \theta) \\ &\quad \times \sum_{\mathbf{x}''} P(\mathbf{x}, \mathbf{x}'', \pi^{gh} | \pi_{x_1} = s_g, \pi_{x_L} = s_h, \theta) \end{aligned}$$

Although it is straightforward to calculate $P(\pi_{x_1} = s_g, \pi_{x_L} = s_h \mid \theta)$, the term becomes problematic in the dynamic programming recursion we define. One or both of the optimal truncation endpoints s_g, s_h are undetermined until the dynamic programming recursion is complete and a traceback is performed. We therefore make our second approximation here, approximating this term as 1.0 when s_g, s_h are consensus match states and 0.0 when they are not. This corresponds to an assumption that all probability mass flows through the consensus match states at the endpoints g, h, neglecting the probability that an SCFG deletion

state could be used at one of these consensus positions. Local alignments will therefore be forced to start and end with consensus match positions (just as in standard Smith/Waterman local sequence alignment). This leaves:

$$P(\mathbf{x}, \pi^{gh} \mid \theta) \simeq \frac{2}{W(W+1)} \sum_{\mathbf{x}''} P(\mathbf{x}, \mathbf{x}'', \pi^{gh} | \pi_{x_1} = s_g, \pi_{x_L} = s_h, \theta)$$

In the next section, we show there is an efficient dynamic programming algorithm for finding the parse tree π^{gh} that performs the necessary summation over missing data and maximizes this joint probability for a given observed sequence fragment $\mathbf{x}$.

2.3.2 Description of the trCYK algorithm

The Cocke-Younger-Kasami (CYK) algorithm is a standard algorithm for calculating the maximum likelihood SCFG parse tree for a given sequence [63, 123, 133, 268]. CYK recursively calculates terms $\alpha_v(i, j)$ representing the log probability of the optimal parse subtree rooted at state v that accounts for a subsequence $x_i \ldots x_j$, initializing at the smallest subtrees and subsequences (model end states aligned to subsequences of length 0) and iteratively building larger subtrees accounting for longer subsequences. At termination, the score $\alpha_0(1, L)$ is the log probability of a parse tree rooted at the model's start state 0 accounting for the complete sequence $x_1 \ldots x_L$. The optimal parse tree is then recovered by a traceback of the dynamic programming matrix. When applied specifically to a CM of M consensus nodes and a sequence of length L, the CYK algorithm requires $O(L^2M)$ memory and $O(L^3M)$ time [70]. A more complex divide-and-conquer variant of the CM CYK algorithm requires $O(L^2 \log M)$ memory [66].

Previously, we implemented subtree-based local RNA structure alignment by a minor adaptation of the CM's generative model that required no substantive alteration of the CYK algorithm. Specifically, we allowed a start transition from the model's root state 0 to *any* of the consensus states in the model, and we allowed an end transition from any consensus state in the model to a "local end" state (EL) that emits zero or more nonhomologous residues with a self-transition loop. The start transition allows the model to align to any model subtree and not just the complete model, and the end transition allows it to replace any subtree with a

nonhomologous insertion.

The truncated sequence local alignment algorithm we describe here, for finding an optimal local parse tree π^{gh} that accounts for a linear sequence fragment, does require a substantial modification of the CYK algorithm because it needs to identify the optimal endpoints g, h. The two approaches to local alignment are not mutually exclusive. We retain the local end transition to an EL state to model nonhomologous replacement of structural elements inside a local parse subtree.

The key property of local parse trees π^{gh} that enables a recursive CYK-style algorithm can be summarized as "once marginal, always marginal", as illustrated in Figure 2.2. As the CYK calculation builds larger and larger subtrees – climbing "up" the model – it will usually grow by adding appropriate (v, i, j) triplets that deal with complete (joint) emission of any base paired residues (upper left panel of Figure 2.2). At some point it may need to decide that the sequence is truncated at either the right or left endpoint of the optimal parse subtree (upper middle and upper right panels of Figure 2.2, respectively), in which case only the left residue x_i or the right residue x_j (respectively) will be added to the growing parse subtree, and scored as the marginal probability of generating the observed residue at state v summed over all possible identities of the missing residue. We refer to this as **joint mode** versus left and right **marginal modes** for a growing alignment. The switch from joint mode to a marginal mode identifies one of the endpoints (h or g, respectively). Once switched, the alignment must stay in that marginal mode until the root state of the optimal local parse subtree is identified. Left marginal mode alignments may only be extended by aligning left residues x_i (central panel of Figure 2.2), and right marginal mode alignments may only be extended by aligning right residues x_j (center right panel of Figure 2.2). In marginal modes, residue emission probabilities involving missing data are the appropriate marginal summation of the state's emission probabilities.

In order to recursively calculate the optimal local parse tree, including these optimal switch points from joint to marginal modes, we extend the CYK algorithm to treat the different modes separately (essentially as an additional layer of hidden state information), and calculate separate matrices for each mode: α^J for the standard case (joint mode), α^L for extension only at the 5' end (emissions are marginalized to the left), and α^R for extension only at the 3' end (emissions are marginalized to the right). Each column in Figure 2.2

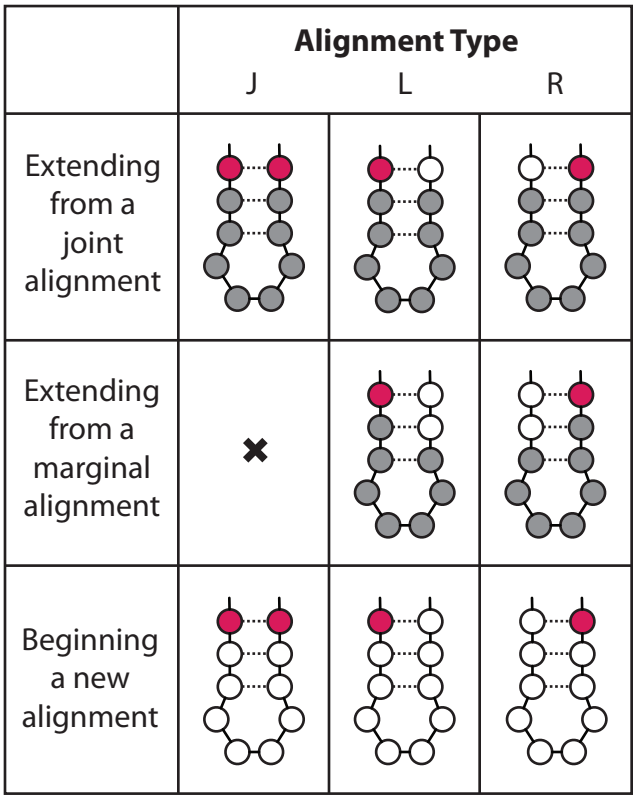


Figure 2.2: **Extension possibilities for building alignments.** Extension may generally be joint (J), left marginal (L), or right marginal (R). **Top:** an existing joint alignment may use any type of extension. Grey circles indicate the previously existing alignment, with the new residues added in red, and open circles for when no residue is aligned. Transitioning from joint to marginal alignment sets an endpoint of the alignment. **Center:** an alignment that is already marginal may only continue with marginal extension on the same side. **Bottom:** a new alignment may be started in any of the three modes. The joint alignment here is shown skipping a portion of the subtree, but that need not be the case. Initiating an alignment in marginal mode also sets one of the alignment endpoints.

illustrates the main cases that have to be examined : for example, the calculation of $\alpha_v^L(i,j)$ for a base-pair emitting state v would examine each of its transition-connected states y and consider both the possibility of reaching (v,i,j) by extension of a previously calculated left-marginal $\alpha_y^L(i-1,j)$, and the possibility of reaching it by switching from a previously calculated joint $\alpha_y^J(i-1,j)$.

The calculation at bifurcation states requires special consideration, as illustrated in Figure 2.3. Only combinations of modes for left and right branches that would form a contiguous subsequence $x_i \ldots x_j$ aligned to a local parse tree rooted at bifurcation state v are allowed. Cases in which an entire branch is missing data must also be considered (shown as $\emptyset$ in Figure 2.3). There is a unique case when both branches of the bifurcation have marginal alignments (bottom of Figure 2.3), and the resulting join cannot be extended further. For convenience, we call the score of this case α^T, noting that it is only defined when v is a bifurcation state and that because it is a terminal case, it does not have to be stored in the recursion.

The score of the optimal local parse tree aligned to a subsequence $x_i \ldots x_j$ is the combination of consensus state v, sequence positions i,j, and mode $X \in \{J,L,R,T\}$ that maximizes $\alpha_v^X(i,j)$. Alternatively, the entire observed sequence $x_1 \ldots x_L$ can be forced into an optimal local alignment to the model by choosing v,X that maximize $\alpha_v^X(1,L)$.

2.3.3 The trCYK algorithm

The following description of the truncated sequence CYK dynamic programming algorithm (trCYK) assumes familiarity with notation and conventions used in previously published descriptions of CMs [63, 66]. Briefly, sequence positions are indexed by i,j and k; x_i is the residue at position i; and d refers to the length of a subsequence $x_i \ldots x_j$ where $d = j - i + 1$. The main parameters of a CM are the emission and transition probabilities of its states. These states are indexed by v,y,z, ranging from 1 to M, the total number of model states. $\mathcal{C}_v$ lists all the *children* y of state v; the transition probability for moving from v to y is $t_v(y)$. (A bifurcation state v splits to y,z with probability 1.0). $\mathcal{S}_v$ is the *type* of state v; possible values are S (start), P (pair emit), L (left emit), R (right emit), D (deletion), B (bifurcation), and E (end). e_v represents emission probabilities at state v, which (depending on the state type) may emit either one or two residues, $e_v(x_i)$ or

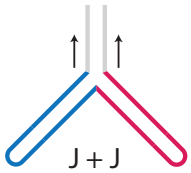
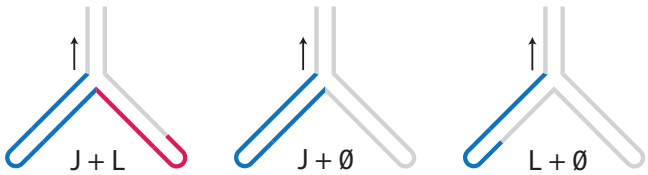
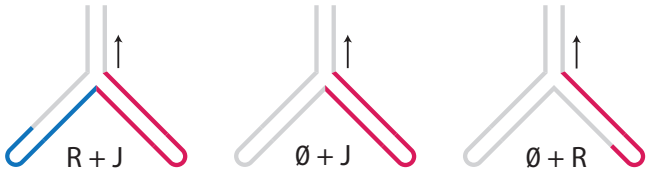
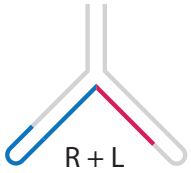
Bifurcation Mode	Possible branch combinations	Next Extension
J		Joint, left marginal, or right marginal
L		Left marginal only
R		Right marginal only
T		Cannot be extended (terminal)

Figure 2.3: **Extension possibilities at bifurcations.** A bifurcation state joins two subtrees, one 5' (left, blue) and one 3' (right, red). Each subtree has its own alignment mode, either joint (J), left marginal (L), right marginal (R), or empty (∅). The subtree modes together must give a continuous subsequence, and all valid combinations are shown. The combination determines the mode of the bifurcation state, which can subsequently be extended like any other state, except for the terminal case T. (Arrows show possibilities for later extension.)

$e_v(x_i, x_j)$. Emission probabilities marginalized over a missing residue are indicated by a $*$ for the missing residue; for example $e_v(x_i, *) = \sum_a e_v(x_i, a)$.

Initialization:

best_score $= -\infty$

for $j = 0$ **to** L, $d = 0$ **to** j

$i = j - d + 1$

$\alpha^J_{\mathrm{EL}}(i, j) = d * \log t_{\mathrm{EL}}(\mathrm{EL})$

$\alpha^{\{L,R\}}_{\mathrm{EL}}(i, j) = -\infty$

for $v = \mathrm{M}$ **down to** 1, $j = 0$ **to** L

if $\mathcal{S}_v = \mathrm{D}$ or S

$\alpha^J_v(j+1, j) \;=\; \max_{y \in \mathcal{C}_v} \left[\alpha^J_y(j+1, j) + \log t_v(y)\right]$

$\alpha^{\{L,R\}}_v(j+1, j) \;=\; -\infty$

else if $\mathcal{S}_v = \mathrm{P}$ or L or R

$\alpha^{\{J,L,R\}}_v(j+1, j) \;=\; -\infty$

else if $\mathcal{S}_v = \mathrm{B}$

$\alpha^J_v(j+1, j) \;=\; \alpha^J_y(j+1, j) + \alpha^J_z(j+1, j)$

$\alpha^{\{L,R,T\}}_v(j+1, j) \;=\; -\infty$

else if $\mathcal{S}_v = \mathrm{E}$

$\alpha^{\{J,L,R\}}_v(j+1, j) \;=\; 0$

Recursion:

for $v = \mathrm{M}$ **down to** 1, $j = 1$ **to** L, $d = 1$ **to** j

$i = j - d + 1$

if $\mathcal{S}_v = \mathrm{D}$ or S

$\alpha^J_v(i, j) \;=\; \max_{y \in \mathcal{C}_v} \left[\alpha^J_y(i, j) + \log t_v(y)\right]$

$\alpha^L_v(i, j) \;=\; \max_{y \in \mathcal{C}_v} \left[\alpha^L_y(i, j) + \log t_v(y)\right]$

$\alpha^R_v(i, j) \;=\; \max_{y \in \mathcal{C}_v} \left[\alpha^R_y(i, j) + \log t_v(y)\right]$

else if $\mathcal{S}_v = \mathrm{P}$

if $d \geq 2$

$$\alpha_v^J(i,j) \;=\; \log e_v(x_i, x_j) + \max_{y \in \mathcal{C}_v} \left[\alpha_y^J(i+1, j-1) + \log t_v(y)\right]$$

$$\alpha_v^L(i,j) \;=\; \log e_v(x_i, *) + \max_{y \in \mathcal{C}_v} \left[\alpha_y^{\{J,L\}}(i+1, j) + \log t_v(y)\right]$$

$$\alpha_v^R(i,j) \;=\; \log e_v(*, x_j) + \max_{y \in \mathcal{C}_v} \left[\alpha_y^{\{J,R\}}(i, j-1) + \log t_v(y)\right]$$

else $\alpha_v^J(i,j) \;=\; -\infty$

$$\alpha_v^L(i,j) \;=\; \log e_v(x_i, *)$$

$$\alpha_v^R(i,j) \;=\; \log e_v(*, x_j)$$

if $\alpha_v^{\{J,L,R\}}(i,j) > \textit{best\_score}$

store $\textit{best\_score}, v, i, j, \textit{mode}$

else if $\mathcal{S}_v = \mathrm{L}$

$$\alpha_v^J(i,j) = \log e_v(x_i) + \max_{y \in \mathcal{C}_v} \left[\alpha_y^J(i+1, j) + \log t_v(y)\right]$$

if $d \geq 2$ $\alpha_v^L(i,j) \;=\; \log e_v(x_i) + \max_{y \in \mathcal{C}_v} \left[\alpha_y^L(i+1, j) + \log t_v(y)\right]$

else $\alpha_v^L(i,j) \;=\; \log e_v(x_i)$

$$\alpha_v^R(i,j) = \max_{y \in \mathcal{C}_v} \left[\alpha_y^{\{J,R\}}(i, j) + \log t_v(y)\right]$$

if $\alpha_v^{\{J,L\}}(i,j) > \textit{best\_score}$

store $\textit{best\_score}, v, i, j, \textit{mode}$

else if $\mathcal{S}_v = \mathrm{R}$

$$\alpha_v^J(i,j) = \log e_v(x_j) + \max_{y \in \mathcal{C}_v} \left[\alpha_y^J(i, j-1) + \log t_v(y)\right]$$

$$\alpha_v^L(i,j) = \max_{y \in \mathcal{C}_v} \left[\alpha_y^{\{J,L\}}(i, j) + \log t_v(y)\right]$$

if $d \geq 2$ $\alpha_v^R(i,j) \;=\; \log e_v(x_j) + \max_{y \in \mathcal{C}_v} \left[\alpha_y^R(i, j-1) + \log t_v(y)\right]$

else $\alpha_v^R(i,j) \;=\; \log e_v(x_j)$

if $\alpha_v^{\{J,R\}}(i,j) > \textit{best\_score}$

store $\textit{best\_score}, v, i, j, \textit{mode}$

else if $\mathcal{S}_v = \mathrm{B}$

$(y, z) = \mathcal{C}_v$

$\alpha_v^J(i,j) \;=\; \max_{i-1 \leq k \leq j} \left[\alpha_y^J(i,k) + \alpha_z^J(k+1,j)\right]$

$\alpha_v^L(i,j) \;=\; \max_{i-1 \leq k \leq j} \left[\alpha_y^J(i,k) + \alpha_z^L(k+1,j)\right]$

$\alpha_v^R(i,j) \;=\; \max_{i-1 \leq k \leq j} \left[\alpha_y^R(i,k) + \alpha_z^J(k+1,j)\right]$

$\alpha_v^T(i,j) \;=\; \max_{i \leq k \leq j-1} \left[\alpha_y^R(i,k) + \alpha_z^L(k+1,j)\right]$

if $\alpha_v^{\{J,L,R,T\}}(i,j) > \textit{best\_score}$

store $\textit{best\_score}, v, i, j, \textit{mode}$

$\alpha_v^L(i,j) \;=\; \max\left\{\alpha_v^L(i,j), \alpha_y^{\{J,L\}}(i,j)\right\}$

$\alpha_v^R(i,j) \;=\; \max\left\{\alpha_v^R(i,j), \alpha_z^{\{J,R\}}(i,j)\right\}$

else if $\mathcal{S}_v = \mathrm{E}$

$\alpha_v^{\{J,L,R\}}(i,j) \;=\; -\infty$

Termination:

$\textit{score} = \textit{best\_score} + \log \frac{2}{W(W+1)}$

After this recursion is completed, the optimal local parse tree may be recovered by traceback from the best scoring $\alpha_v^X(i,j)$. To facilitate this, it is helpful to store traceback pointers during the dynamic programming recursion; for clarity, these are not indicated in the algorithm description above. In order to avoid parsing ambiguity, any ties in the traceback are resolved in favor of joint mode over marginal modes. Thus marginal mode is only used when required to account for one or more missing residues in the local parse tree.

It is worth noting that an additional kind of structural alignment locality is dealt with by the state structure of a covariance model, rather than by the trCYK algorithm. The alignable subsequence (as identified by trCYK) may also be subject to large internal deletions and insertions relative to the consensus RNA structure. CMs accommodate large structural insertions and deletions by allowing any consensus state to transition to a special "EL" (end-local) state which has a self-transition loop, thereby allowing any structural domain to be truncated anywhere and replaced by zero or more nonhomologous residues. The EL state appears in the recursion above, and its use and rationale for accommodating local structural variation are more fully explained elsewhere [140].

2.4 Implementation

The algorithm described above requires $O(L^2M)$ memory to store traceback pointers for recovering an optimal local parse tree. In order to be able to align large RNAs, we also implemented an extension of the divide and conquer approach described in [66] to trCYK, reducing the memory requirement to $O(L^2 \log M)$ at the expense of a small increase in computation time. The divide and conquer version was used to obtain the results described below. Both versions are provided in the ANSI C source code of Infernal in the supplementary material.

trCYK has an upper bound time complexity of $O(L^3M)$, the same as standard CYK. trCYK's additional calculations and three matrices in place of one contribute a constant multiplier. Empirically, trCYK runs about five-fold slower than standard CYK on the same problem size. For example, a single RNase P alignment for the results in Figure 2.2 (283 nodes and 1119 states in the model; sequence length of 400) took 40 seconds for `trcyk` vs. 9.4 seconds for Infernal `cmsearch` with equivalent settings, on a single 3.0GHz Intel Xeon processor.

2.5 Evaluation

We compared the effectiveness of the trCYK method for local structural RNA alignment to the previous subtree method, by measuring how accurately and completely the two methods align single shotgun sequence reads to structural RNA consensus models. To do this, we constructed a synthetic test of realigning simulated reads generated by sampling sequence fragments from trusted (presumed correct) alignments. We chose to use simulated data instead of real data because we are interested in conducting a controlled comparison of the two algorithms against known correct answers. Because alignment quality and (in particular) local alignment coverage are strongly affected by parameterization, in order to isolate the algorithm's effect, we used the same profile SCFGs as parameterized by the same implementation (Infernal), and compared Infernal's default subtree alignment method versus its newly implemented trCYK option. To put this comparison in context, we also test two other primary sequence methods: pairwise alignment with BLASTN, and sequence profile

alignment with HMMER.

We used multiple sequence alignments of two well known structural RNA genes, small subunit ribosomal RNA (SSU rRNA) and RNase P. SSU rRNA is in general highly conserved, so in many regions of the RNA consensus and for all but the most outlying taxa, SSU rRNA is usually not a particularly challenging sequence alignment problem. RNase P sequences, in contrast, tend to be highly divergent at the primary sequence level. For a trusted RNase P multiple alignment, we used the bacterial class A seed alignment from Rfam 8.1 [26, 100], and our trusted SSU rRNA alignment was adapted from the bacterial seed alignment from the Comparative RNA Web Site, CRW [35]. Due to the large number of sequences, the SSU alignment was filtered to remove sequences such that no aligned pair was more than 92% pairwise identical. It was also edited slightly towards a consensus structural alignment in preference to an evolutionarily correct alignment where there was ambiguity between structural conservation and homology. Our SSU rRNA alignment is provided in the supplementary material.

The sequences in the trusted alignment were clustered by single linkage by pairwise identity (as defined by the original alignment) and split into a smaller training alignment subset and a testing set, so as to minimize pairwise identity between training and test data and create more challenging alignment tests. For SSU, this gave 101 training sequences and 51 testing sequences, with maximum identity between sets of 82%. The smaller RNase P family has 28 training sequences and 15 testing sequences, with maximum identity of 60%.

We simulated a genomic context for each test sequence, consisting of randomly generated sequence of the same monoresidue composition, and then sampled a random subsequence of length 800 (SSU rRNA) or 400 (RNase P) that contained at least 100 nucleotides of the RNA. Five fragments were sampled for each SSU rRNA test sequence, and ten for each RNase P test sequence, for a total of 255 SSU test fragments and 150 RNase P test fragments. The 800 nt length of SSU rRNA test fragments roughly corresponds to the average single read length in recent metagenomic sequencing surveys with Sanger sequencing technology [212, 244]. RNase P is a shorter RNA (average length just 310 nt) so shorter fragments of 400 bases were used, roughly according to the current capabilities of newer 454 sequencing technology. The training alignments and test fragments are provided in the supplementary material.

We aligned each test sequence fragment to the training alignment using four different local alignment methods. For BLAST local sequence alignment, we used NCBI blastn [7], with default parameters except for a word length of W=6 and an E-value cut-off of 1.0, and used the pairwise alignment with the lowest E-value to identify the nearest neighbor among any of the individual training set sequences. All alignments to that nearest-neighbor, including lower scoring ones, were considered as portions of the complete alignment. For profile alignment, we used HMMER 2.3.2 [67] to build a profile HMM from the training alignment subset with `hmmbuild` using the `-f` option to build local alignment models, and aligned each test fragment to the profile (thereby adding it to the multiple alignment with the training sequences) with `hmmalign` using default parameters. For the subtree-based local alignment method, we used Infernal version 1.0 to build a CM of the training alignment subset with `cmbuild` with the `--enone` option to shut off entropy weighting. (We have observed that Infernal's entropy weighting option [185] is appropriate for maximal sensitivity in remote homology search, but not for alignment accuracy; DLK, SRE, and Eric Nawrocki, unpublished observations.) We aligned each test fragment to the CM with `cmsearch` using default parameters. Finally, for trCYK, we used the `trcyk` program included in Infernal 1.0 to align test fragments to the same CM used for `cmsearch`.

To evaluate the alignments, they are mapped to the reference alignment using an intermediary sequence; for blastn, this is the nearest-neighbor sequence it was aligned to, and for the profile methods it is the consensus sequence of the model. If a residue in the test sequence was aligned to a non-gap position in the reference alignment, it is correct in the output alignment if it is aligned to that same position, and incorrect otherwise. If the residue was originally aligned to a gap position, it is judged to be correct if, in the output alignment, it is between the same two consensus positions that bordered the original gap. Misaligned residues include both residues that should be aligned but are incorrect, and residues that should not be aligned at all (part of the surrounding "genomic" sequence). We measured both the accuracy of the resulting alignments (positive predictive value, PPV: the fraction of aligned positions that are also found in the trusted alignment) and the coverage (sensitivity: the fraction of aligned positions in the trusted alignment that are found in the calculated local alignment).

The results, mean and standard deviations for sensitivity and PPV for each method, are shown in Figure

2.4. BLAST generally returns highly accurate alignments, but has low coverage, corresponding to a tendency to pick out only the most highly conserved portions of the true alignments. (The default NCBI BLAST scoring scheme is tuned for high sequence identity. In principle we should be able to improve the coverage somewhat by a different choice of scoring matrix.) Profile HMMs (hmmalign) achieve both high accuracy and coverage. CMs with subtree-based local alignment (cmsearch) show poor coverage relative to HMMs, illustrating the issue that motivated this work. The new method, trCYK, matches the coverage of profile HMM sequence alignment, while providing higher accuracy. The improvement is not large, but even small increases in coverage and accuracy are important when the alignment is to be used in downstream phylogenetic analyses that are sensitive to both.

2.6 Discussion

The trCYK algorithm performs local structural RNA alignment in a manner that uses secondary structural information (correlated base pairs) where possible, and reverts to sequence alignment when a truncation has removed sequence that would be base paired. Alignment coverage of sequence fragments (such as single reads from metagenomic shotgun sequencing) is maximized, while still retaining the accuracy of CM-based RNA structural alignment methods. The trCYK algorithm rests on good theoretical ground by viewing the sequence truncation problem formally as a missing data inference problem, and it makes only two minor assumptions to simplify the missing data inference problem to one that can be solved by a relatively efficient dynamic programming recursion.

The disadvantage of trCYK is that unlike local primary sequence alignment, which is as computationally efficient as global sequence alignment, it needs to track the three possible structural alignment modes (joint, left marginal, and right marginal) in the dynamic programming recursion. This imposes about a three-fold increase in memory and five-fold increase in CPU time required relative to previous CM alignment implementations. This cost is unfortunate, as the use of CM-based approaches is already limited by their relatively high computational complexity. We expect to be able to accelerate trCYK with the same approaches we are

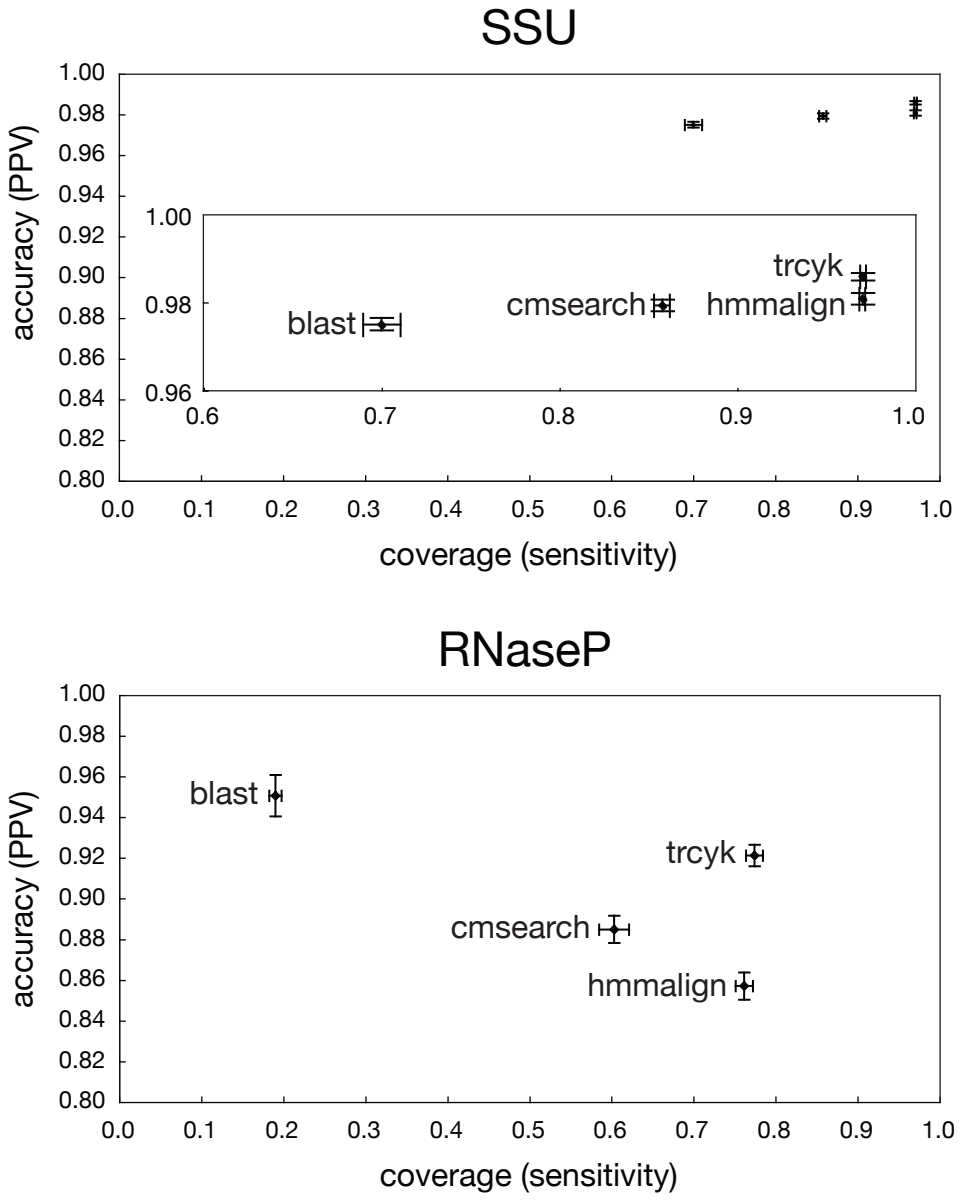


Figure 2.4: **Per-residue accuracy of alignment methods.** Alignment of simulated metagenomic reads compared against a reference alignment for four alignment methods: primary sequence (blastn), primary sequence profile (hmmalign), CM with subtree-based local alignment (cmsearch), and CM with truncated sequence model (trcyk). Means and standard deviations for sensitivity and PPV are plotted. Top: alignment of 800nt fragments to the bacterial small subunit ribosomal RNA. Bottom: alignment of 400nt fragments to bacterial RNase P.

developing for standard CYK using the subtree-based alignment model [185]. We additionally expect it will be feasible to develop simple accelerated heuristics for identifying optimal or near-optimal switch points from joint to marginal alignment modes, in order to bypass the need for full dynamic programming. For example, we should be able to use fast primary sequence alignment to determine likely endpoints of the alignment on the consensus yield of the structural model, and from that derive a CM with an appropriately marginalized partial structure. We therefore envision trCYK's future role as a rigorous baseline against which more heuristic local RNA structural alignment methods may be compared.

2.7 Acknowledgments

This work was supported by a National Science Foundation Graduate Research Fellowship to DLK, and by the Howard Hughes Medical Institute.

2.8 Divide and Conquer extension of trCYK

The full trCYK algorithm as presented in 2.3.3 is memory intensive; prohibitively so for all but the smallest cases. If only the score is needed, this can be calculated efficiently by discarding alignment information as soon as it is not needed for later score calculations. However, usually an alignment is needed in addition to the score, in which case traceback information must be stored. The size of the traceback matrix is $\mathcal{O}(L^2M)$ and even models and sequences of moderate sizes quickly exceed the capacity of memory. For practical application, a divide-and-conquer approach to split the problem into manageable subsections is essential. The approach here closely follows that of Eddy [66], with modifications for the previous model of local alignment as well as those required for adaptation to trCYK.

To achieve this, we need a complementary set of values (denoted β), such that the sum of α and β for any set of indices (v, i, j) is the score of the best alignment that includes that triplet in its trace. Then, considering a single state or set of states v through which all traces are expected to pass exactly once, maximizing $\alpha + \beta$ over all (i, j) provides the total score of the best possible alignment, as well as set of values (v, i, j) that it must use. The problem can then be split into new smaller problems, using these values as new endpoints. The algorithm for calculating β is trCYK/outside, whereas the trCYK described previously may be called trCYK/inside for clarity. These algorithms operate in the same direction as the Outside and Inside algorithms respectively; however, they still calculate the best path rather than integrating over all paths. In addition, we will need a small collection of methods for determining how to split a particular type of problem.

2.8.1 trCYK/inside

Some small changes to trCYK are required in order to incorporate it into a divide-and-conquer framework. Most of these relate to the cases where trCYK/inside is only called to solve a defined subset of the problem, rather than find a complete solution.

First, we define a sequence range (g, q) which limits the scope of the sequence positions i and j. Aligning a complete sequence is equivalent to setting $g = 1$ and $q = L$. Second, we may also have a limited range of

model states to consider, (r, z) rather than the complete model $(1, M)$. By convention, the range (r, z) may be either a complete subtree from a state r to last leaf node z, or it may be an unbifurcated linear range of model states. This state r is considered the local root of the subtree, but it is generally not the same as the global ROOT state of the model.

The second case, where the state range under consideration ends at a state not guaranteed to be a leaf node of the model tree, is a special type of problem which we refer to as 'V-type' (see also 2.8.3). In this case, there is a 'hole' represented by the state and sequence triple (z, h, p) which can be assumed to already have been aligned, and further alignment proceeds from that point to the triple (r, g, q), which completely encloses it. (That is, $g \leq h$ and $p \leq q$, and $r..z$ represents a continuous, unbifurcated branch of the model.) These g and h indices are unrelated to those used for local parse trees in 2.3.1 ; they are used here for notational consistency with the previous description of Divide and Conquer for CYK [66].

When subdividing the problem, we must keep track of whether endpoints of the alignment have already been set outside of the current subproblem. If one or both endpoints have been determined, we must ensure both that no new endpoints are introduced, and that the alignment extends as far as is necessary to connect with the other pieces of the alignment.

Finally, the penalty for aligning to a fragment rather than the entire model, $\log \frac{2}{W(W+1)}$, is pulled out of trCYK/inside to an external wrapper to ensure that it is added only once, because trCYK/inside is generally called many times in the solution of a single alignment.

2.8.2 trCYK/outside

The trCYK/outside algorithm iteratively calculates $\beta_v(i, j)$, the log probability of the maximum likelihood alignment of a sequence (g, q) to a model, excluding the subsequence (i, j) aligned to the portion of the model rooted at v. That is, $\beta_v(i, j)$ is the score for the complete alignment of (g, q) to a model starting with state r, except the portion that would be covered by $\alpha_v(i, j)$. The description given here represents of a simplification of the more general algorithm: we assert that the model structure contains no bifurcations between states r and v, and the algorithm will only be applied when those conditions are met.

Like for α in trCYK/inside, β is augmented with marginal modes to allow the alignment of partial sequences. $\beta_v^J(i,j)$ asserts that the parent state of v was in joint mode. Likewise, β_v^L and β_v^R indicate that the parent state of v was in left or right mode, respectively. Looking at $\beta_v^R(i,j)$, note that because the parent of v is right marginal, all further parents up to the beginning of the alignment must be right marginal. Thus, $\beta_v^R(i,j)$ cannot have emitted any sequence on the left margin in its construction - the subsequence $(g, i-1)$ is guaranteed to be excluded from the alignment, for any value of i, as well as the subsequence (i,j), which is excluded by definition. Therefore, the notation can be simplified to $\beta_v^R(j)$, indicating that the alignment excludes all subsequence up to and including j. Similarly, the left marginal calculation simplifies to $\beta_v^L(i)$. Analogous to $\mathcal{C}_v$, the set of children of state v which we used in trCYK/inside, we also need $\mathcal{P}_v$ for the parents of state v, that is, the set of states which have transition paths to state v. $\delta(v, \mathrm{X})$ is a delta function on whether the combination of state v and mode X is a valid endpoint of the alignment – score 0.0 if true, $-\infty$ otherwise.

Initialization:

for $i = \mathrm{g}$ **to** q

$\beta_{\mathrm{EL}}^L(i) = -\infty$

$\beta_r^L(i) = \delta(r, L)$

$\beta_r^L(\mathrm{g}-1) = -\infty$

for $j = \mathrm{q}$ **down to** g

$\beta_{\mathrm{EL}}^R(j) = -\infty$

$\beta_r^R(j) = \delta(r, R)$

$\beta_r^R(\mathrm{q}+1) = -\infty$

for $j = \mathrm{q}$ **down to** $\mathrm{g}-1$; **for** $i = \mathrm{g}$ **to** $j+1$

$\beta_{\mathrm{EL}}^J(i,j) = -\infty$

$\beta_r^J(i,j) = \delta(r, J)$

$\beta_r^J(\mathrm{g}, \mathrm{q}) = 0$

Recursion:

for $v = \mathrm{r}+1$ **to** z

$\beta_v^L(\mathrm{g}-1) = -\infty$

$\beta_v^R(\mathrm{q}+1) = -\infty$

for $i = \mathrm{g}$ **to** q

$$\beta_v^L(i) = \max_{y \in \mathcal{P}_v} \begin{cases} \delta(v, L) & \\ \beta_y^L(i) + \log t_y(v) & : \mathcal{S}_y = \mathrm{R}, \mathrm{D}, \mathrm{S}, \mathrm{E} \\ \beta_y^L(i-1) + \log t_y(v) + \log e_y(x_{i-1}) & : \mathcal{S}_y = \mathrm{L} \\ \beta_y^L(i-1) + \log t_y(v) + \log e_y(x_{i-1}, *) & : \mathcal{S}_y = \mathrm{P} \end{cases}$$

for $j = \mathrm{q}$ **down to** g

$$\beta_v^R(j) = \max_{y \in \mathcal{P}_v} \begin{cases} \delta(v, R) & \\ \beta_y^R(j) + \log t_y(v) & : \mathcal{S}_y = \mathrm{L}, \mathrm{D}, \mathrm{S}, \mathrm{E} \\ \beta_y^R(j+1) + \log t_y(v) + \log e_y(x_{j+1}) & : \mathcal{S}_y = \mathrm{R} \\ \beta_y^R(j+1) + \log t_y(v) + \log e_y(*, x_{j+1}) & : \mathcal{S}_y = \mathrm{P} \end{cases}$$

for $j = \mathrm{q}$ **down to** $\mathrm{g}-1$; **for** $i = \mathrm{g}$ **to** $j+1$

$$\beta_v^J(i,j) = \max_{y \in \mathcal{P}_v} \begin{cases} \delta(v, J) & \\ \beta_y^J(i,j) + \log t_y(v) & : \mathcal{S}_y = \mathrm{D}, \mathrm{S}, \mathrm{E} \\ \max \begin{bmatrix} \beta_y^J(i-1, j+1), \\ \beta_y^L(i-1), \\ \beta_y^R(j+1) \end{bmatrix} + \log t_y(v) + \log e_y(x_{i-1}, x_{j+1}) & : \mathcal{S}_y = \mathrm{P} \\ \max \begin{bmatrix} \beta_y^J(i-1, j), \\ \beta_y^L(i-1), \\ \beta_y^R(j) \end{bmatrix} + \log t_y(v) + \log e_y(x_{i-1}) & : \mathcal{S}_y = \mathrm{L} \\ \max \begin{bmatrix} \beta_y^J(i, j+1), \\ \beta_y^L(i), \\ \beta_y^R(j+1) \end{bmatrix} + \log t_y(v) + \log e_y(x_{j+1}) & : \mathcal{S}_y = \mathrm{R} \end{cases}$$

for $j = \mathrm{q}$ **down to** $\mathrm{g}-1$; **for** $i = \mathrm{g}$ **to** $j+1$

$d = j - i + 1$

$$\beta^J_{\mathrm{EL}}(i,j) = d * \log t_{\mathrm{EL}}(\mathrm{EL}) + \max_{\mathrm{r} \le v \le \mathrm{z}} \begin{cases} \beta^J_v(i-1,j+1) + \log t_v(\mathrm{EL}) + \log e_v(x_{i-1}, x_{j+1}) & : \mathcal{S}_v = \mathrm{P} \\ \beta^J_v(i-1,j) + \log t_v(\mathrm{EL}) + \log e_v(x_{i-1}) & : \mathcal{S}_v = \mathrm{L} \\ \beta^J_v(i,j+1) + \log t_v(\mathrm{EL}) + \log e_v(x_{j+1}) & : \mathcal{S}_v = \mathrm{R} \\ \beta^J_v(i,j) + \log t_v(\mathrm{EL}) & : \mathcal{S}_v = \mathrm{D}, \mathrm{S}, \mathrm{E} \end{cases}$$

Termination:

$$\textit{best\_score} = \max_{\substack{\mathrm{r} \le v \le \mathrm{z} \\ \mathrm{g} \le j \le \mathrm{q} \\ \mathrm{g} \le i \le j+1}} \begin{cases} \beta^J_{\mathrm{EL}}(i,j) & \\ \beta^L_v(i) + \log e_v(x_i, *) & : \mathcal{S}_v = \mathrm{P} \\ \beta^L_v(i) + \log e_v(x_i) & : \mathcal{S}_v = \mathrm{L} \\ \beta^R_v(j) + \log e_v(*, x_j) & : \mathcal{S}_v = \mathrm{P} \\ \beta^R_v(j) + \log e_v(x_j) & : \mathcal{S}_v = \mathrm{R} \end{cases}$$

We check β^L and β^R for high-scores, since they may represent complete alignments, after adding the emission score for the final aligned state. β^J must be combined with a corresponding α^J in order to form a complete alignment, so we don't check those scores individually, aside from the case where EL is used before reaching the split point that we are examining. Although this level of detail has been omitted for clarity, the assumptions that alignment endpoints must correspond to sequence-emitting consensus states, rather than insertions, deletions, or non-emitting states, are again enforced here.

2.8.3 Problem division methods

Subdivision of the problem depends on being able to identify a state or set of states which an alignment must use. To accomplish this, we take advantage of known features of the CM architecture. One type of division occurs at bifurcation states, which must be utilized to align to a branching structure. The other type of division uses the node structure of the model: for each consensus position in the model, any alignment

must pass through exactly one match state or delete state. These three subdivision methods below are directly adapted from those used by Eddy [66], as illustrated in figure 2.5.

Generic splitter

The generic splitter divides the alignment problem at bifurcations. It identifies the first bifurcation state v in the model, and its two child states w and y. trCYK/inside calculates α for the trees rooted at w and y, and trCYK/outside calculates β from r down to v. Then, for all sets of sequence positions (i, k, j), we search for the best combination of $\beta_v(i, j) + \alpha_w(i, k) + \alpha_y(k + 1, j)$ where α_w covers the sequence from i to k, α_y covers the sequence from $k + 1$ to j, and β_v covers the divided sequence outside of (i, j) to the ends. A variety of mode combinations are possible, following the same rules as trCYK/inside at B states. This means that of the three expected subproblems (parent, left child, right child), one may represent an empty set, aligning a zero-length sequence to zero states. The child problems w and y can be solved by recursive application of the generic splitter. The portion of the tree from r to v is an unbifurcated linear range of states, aligned to a divided sequence with a hole of intervening sequence. We call this a V-type problem, and it is handled by the V splitter method.

For fully local alignment, we also address a small number of other possible cases; ways in which it might be possible to skip the bifurcation state entirely. For example, the correct alignment may be entirely within one of the two child subtrees, not reaching the bifurcation state. For this, we track the best local score encountered in the calculation of the α and β matrices, to see if any of them individually surpass the total score of the best alignment passing through the bifurcation state. Also, if the model structure given to this method has no bifurcations, it is re-classified as a wedge-type problem and transferred to the wedge splitter method.

Wedge splitter and V splitter

The wedge splitter is used when the remaining portion of the model to be aligned contains no bifurcations, and covers a contiguous subsequence, but still requires further subdivision. The point at which to split the problem

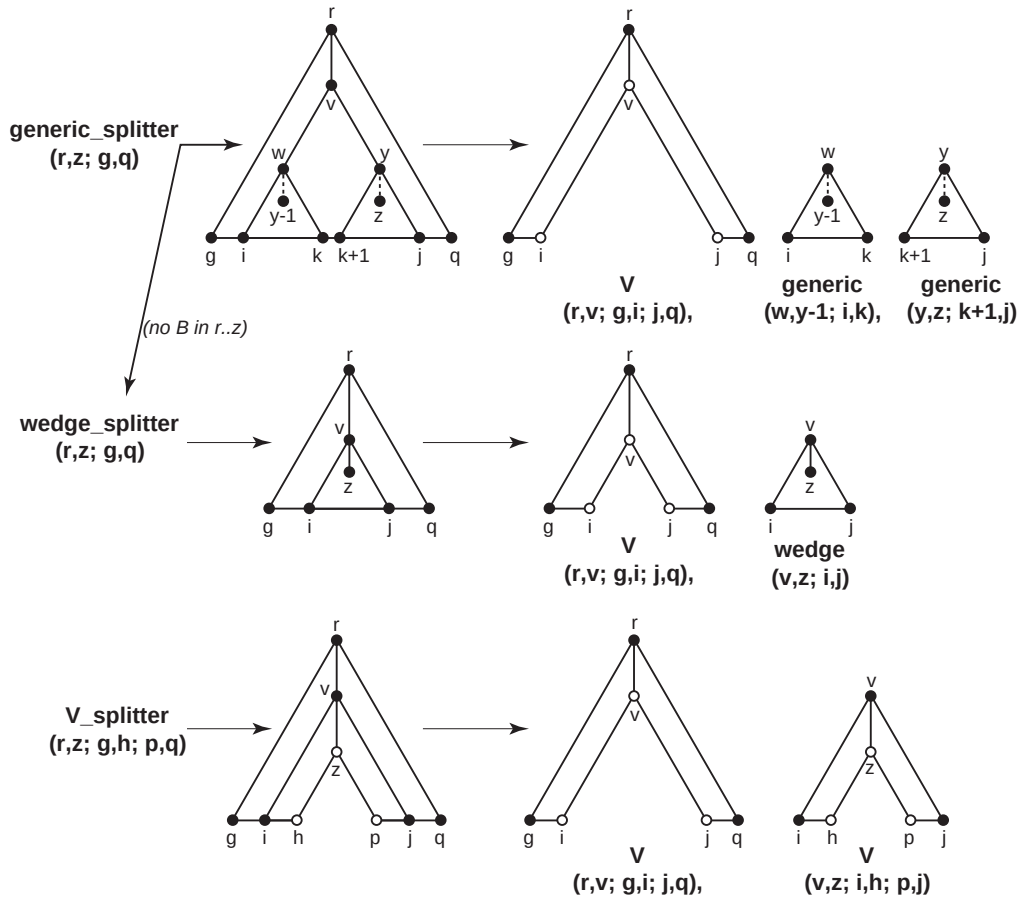


Figure 2.5: **Three problem division methods.** The sequence axis (e.g. $x_g..x_q$) is horizontal. The model subgraph axis for a contiguous set of states (e.g. states r..z) is vertical, where a solid line means an unbifurcated model subgraph, and a dashed line means a model subgraph that may contain bifurcations. Closed circles indicate "inclusive of", and open circles indicate "exclusive of".

Figure is reproduced from Eddy, 2002 [66], under the Open Access license, and is available from `http://www.biomedcentral.com/1471-2105/3/18`.

is determined by finding the match state closest to the middle, and considering that state, its alternative match states (if any), and the corresponding delete state as possible split points. Generally this produces a V-type subproblem from the root to the split point, and a wedge-type subproblem from the split point down, although one of these may be empty. Like the case of the generic splitter, we may also consider alignments that are local and entirely above or below the split point without passing through it.

The V splitter operates very similarly to the wedge splitter, with the exception that it operates on a split subsequence, excluding some intervening interval, rather than a contiguous subsequence. The two subproblems produced from this type of a split are both also V-type problems. These subdivision methods are applied recursively until reaching a point where the size of the remaining problem can easily be stored in memory, at which point it maybe solved by application of trCYK/inside with traceback.

Chapter 3

Statistical significance estimates under trCYK

3.1 Abstract

Estimates of the statistical significance of alignments are based on the distribution of scores of alignments from random sequences. Local alignment scores are observed to conform to an extreme value distribution (EVD) with the two parameters μ and λ. For ungapped local alignments of sequences, analytical results show that λ has a constant value of $\ln(b)$, where b is the base of the logarithmic scoring system. Arbitrary gapped alignments do not follow this behavior; however, it has been conjectured that a fully probabilistic model of sequence alignment would produce a constant lambda, and this is supported by recent empirical results. trCYK is a method of local alignment for probabilistic covariance models of sequences with secondary structure. Because this model shares some important features of the sequence-only alignment model that exhibits constant lambda, it was conjectured that trCYK might also share that behavior. However, I demonstrate that the current probabilistic model for sequence-structure alignment does not have the property of constant lambda, and further work will be necessary to determine the source of the discrepancy.

3.2 Introduction

When searching a database for homologous sequences, it is important to consider not only the alignments that are found, but also the statistical significance of those alignments. Although clear outliers may be easily identified if there are a few scores many times better than the bulk of the distribution, as we lower our score threshold to increase sensitivity, there is increased risk that the alignments may represent chance similarities. Furthermore, as database size increases, the probability of randomly finding high-scoring alignments increases just by virtue of the number of comparisons being performed. Thus it is crucial that our search programs be able to provide an accurate assessment of the likelihood that an alignment was produced by chance, usually expressed as either a P-value (the probability that a score this high or greater was produced by a random, non-homologous sequence), or an E-value (the number of random alignments of this score or greater expected to be encountered, given the size of the database search).

Although they were initially modeled as a normal distribution, it has long been known that the scores for local alignments of random sequences conform to a type I Extreme Value Distribution (EVD, also known as Gumbel), which has slow decay in the right-hand tail compared to normally distributed data. In its general form, the probability that a score S is at least as large as some threshold x may be expressed as

$$P(S \geq x) = 1 - \exp\left[-e^{-\lambda(x-\mu)}\right]$$

with parameters μ and λ controlling location and scale, respectively. These parameters can vary depending on the algorithm, the scoring scheme, and the length and composition of the query and target sequences, and typically must be fit empirically for a given set of conditions. In the most direct approach, the distribution may be fit by simulating and aligning a large number of random sequences, assigning maximum likelihood estimates of the parameters based on the observed scores.

The parameter λ describes the rate of decrease in the likelihood of a score as those scores increase, and it is inside a double exponential. As such, even a small error in the value of λ can result in large errors in the significance estimate when considering the scores most likely to be of interest (that is, scores with low probability of occurring by chance). Thus, calibration typically must simulate and score thousands or tens

of thousands of random sequences in order to calculate λ with high precision and provide accurate E-value estimates. These calibration simulations represent a quite large overhead for searches, and must be performed separately for each and every scoring system. For query-independent scoring systems, using substitution matrices and affine gap models, these statistics are pre-computed for commonly used score parameters [7]. However, for the profile-based search of hidden Markov models (HMMs) or covariance models, each query represents a different scoring scheme, and must be individually calibrated.

Because of the time-intensive nature of the simulations, much work has been directed at improving speed and accuracy of parameter estimates. One major development, island statistics, captures additional score distribution information per comparison of random sequences, by tracking the scores of those sub-optimal alignments that are independent of the highest scoring alignment [4, 270]. This methods allows a trade-off between speed and precision, either producing better estimates with the same size of simulation, or maintaining the accuracy of estimates with a smaller, faster simulation. An alternative approach, importance sampling, weights the generation of random sequences towards those cases which contribute to the high-scoring tail - those rare events which produce large scores by chance [29]. By oversampling high scoring alignments in a controlled fashion, it is possible to re-weight these observed events according to their relative distribution under normal sequence generation and back-calculate the ordinary score distribution. This minimizes the time spent calculating alignments of sequences for which there is no very high scoring pair, and thus little contribution to the cumulative score distribution [29]. Despite these advances, calibration of score distributions remains computationally intensive.

Although much work has gone into improving empirical estimates for λ, determining these parameters by simulation is not always necessary. It has long been known that for local *ungapped* alignment, λ can be directly calculated as the unique positive solution to the equation

$$\sum_{a,b} f_a f_b e^{\lambda s_{a,b}} = 1$$

where f_a is the frequency of residue a, and $\sigma_{a,b}$ is the score for aligning residue a to residue b [3]. Because λ here is simply a multiplier on the scoring matrix, it may be set to any convenient value by applying compensatory scaling to the matrix. Frequently, $\lambda = \ln 2$, explicitly transforming the scoring matrix into units of

bits, equivalent to using base 2 when deriving the scoring matrix via log-odds ratios.

This result, where λ is known *a priori*, has only been conclusively shown to hold for the case of ungapped alignment, and it is known to be untrue for alignments with arbitrary affine gap penalties. However, it has been conjectured that it might extend to certain models of gapped alignment, either in the limit of infinitely long sequences [29], or in the case of fully probabilistic alignment [68].

Several key changes introduced in the development of HMMER3 have drastically simplified the estimation of score distribution parameters for HMMs [68]. The first of these differences is the inclusion of explicit states modeling the non-homologous residues in a sequence. The states allow the model, and thus the score, to be for the entire sequence, rather than only the portions which are homologous; and their length distributions can be adjusted to match the length of the observed sequence. The second change is an implicit model of fragment generation, that is, a probabilistic model covering the start and end points for alignments of homologous segments. These two features, in conjunction with myriad smaller changes, produced an interesting set of observed behaviors of λ. For Viterbi scores (maximum likelihood alignments), λ clusters closely around $\ln(2)$; and Forward scores (total likelihood scores), although they do not conform to a Gumbel distribution, have an exponential tail which also converges to $\lambda = \ln(2)$. The inclusion of an explicit length model of the target sequence means that scores no longer scale depending on sequence length, but rather can be considered independent of target length. The length independence is important for quick calculation of score significance because although the location parameter μ could be adjusted based on target length under Karlin-Altschul statistics for Viterbi scores, no such transformation was known for Forward scores [68, 132].

Stochastic context-free grammars (SCFGs), of which covariance models (CMs) are an example, have many parallels with HMMs. SCFGs introduce an additional level of complexity, allowing long-range interactions and a branching tree structure rather than the linear arrangement of HMMs. This provides a good analog for the secondary structure of folded RNAs, although notably it does not capture non-nested base pairs (pseudoknots) or higher order interactions such as base triples. The similarity between HMMs and CMs extends to both the model structure, which in both cases is based on insertions and deletions relative to a consensus, and to the algorithms for alignment, which operate on broadly similar principles. Historically,

advances developed for HMMs have later been applied to CMs as well, both in model parameterization (e.g., Dirichlet priors [185]) and in algorithms (e.g., memory-efficient Divide-and-Conquer [66]).

trCYK is another example of the adoption by CMs of methods previously used for HMMs. Local alignment with the trCYK method permits incomplete fragments, such as those that might be seen in unassembled metagenomic sequencing, while retaining as much secondary structure information as possible. As part of its model for the generation of such fragments, trCYK incorporates the implicit uniform distribution of fragments model and the sequence length model used by HMMER3 [68]. Although numerous changes were made between HMMER2 (which does not exhibit a constant λ behavior) and HMMER3, the implicit fragment distribution is thought to be one of the key factors in producing this effect. Thus, I had theorized that by adopting this fragment model, trCYK alignment for CMs might also inherit the constant λ behavior.

3.3 Methods

The simulations described here were done using a utility program called `trcyk_fit`, which is distributed with the trCYK code in the Infernal software package. It accepts as its single argument a CM file for which the score distribution is to be estimated. Optional parameters include the number and length of sequences to be used in the simulation. Sequences are generated according to an independent and identically distributed (i.i.d.) model and uniform residue composition. The sequences are aligned under trCYK as described in chapter 2, and the collected scores are fit to an extreme value distribution. The output provides plots of the surviving fraction versus score for both the simulated sequences and the EVD fit to those data, as well as the maximum likelihood estimate λ and μ parameters.

CMs were built with entropy-weighting turned off for the initial experiments, as low entropy is observed to increase the variation in lambda estimates. The local end (EL) model for large deletions, insertions, or non-homologous replacements was also turned off, as the overall probability setting for use of EL had a strong effect on λ estimates, and I was concerned that this would confound the analysis.

HMMs were built with HMMER3 3.0 beta 3, with entropy-weighting also off. Score distributions for

HMMs were estimated with `hmmsim`, which was modified to expect RNA rather than protein alignments.

In the simulations here, 10000 sequences of constant length 500 were sampled for each family in Rfam 9.1 [85], with the exception of a few very large families that were excluded because of running time. The length 500 is an order of magnitude larger than the largest alignments generally seen from random sequences, so edge effects are not expected to be a major consideration. Alignments from HMMER3 were limited to single-hit mode, because trCYK operates exclusively as single-hit alignment.

3.4 Discussion

A summary of maximum likelihood estimates for λ over Rfam families is shown in figure 3.1. The distribution of λ for trCYK is suspiciously close to $\ln(2)$, but it has far too much variation to be considered effectively constant, and is noticeably more variable than HMMER3 in the same experiment. (Note that for all of these simulations, the experiments are shorter than what would be required for full convergence of λ under HMMER3; this is constrained by the execution time of trCYK. Presumably, λ estimates under trCYK would also converge somewhat more with longer experiments, but I believe the relative performance in these shorter simulations is sufficiently informative.)

The observed variation cannot be explained simply by sampling noise; the variation in λ values between experiments is much smaller than the variation between families (data not shown).

This disparity called into question whether trCYK correctly implemented the fragment generation model and the length distribution model which it had adapted from HMMER3. (Those two elements are probably the largest factors affecting the observed constant λ phenomenon.) To test this, the simulations were run again, but this time with all secondary structure annotation removed from the Rfam models. In the absence of secondary structure, CMs treats alignments as purely linear sequences, effectively reducing to a profile-HMM. Although the details of model structure and parameterization are slightly different, the behavior of trCYK and HMMER3 should be nearly identical in this case.

The results for this test are given in figure 3.2, and show that lambda estimates are vastly closer to constant

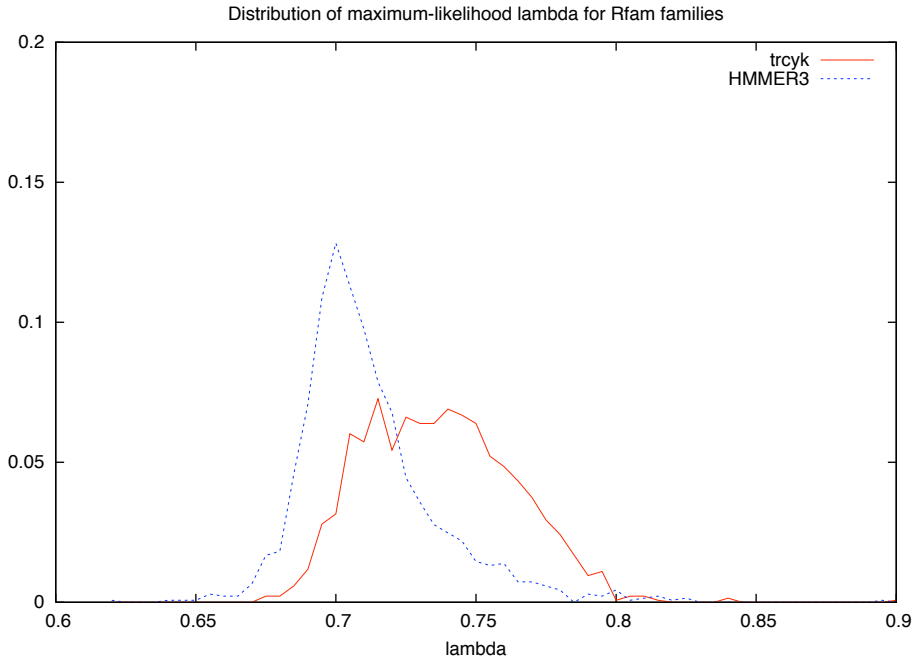


Figure 3.1: **Distribution of lambda for trCYK and HMMER3.** Maximum-likelihood fit of lambda for score distributions under trCYK and HMMER3 alignment. Simulations of 10000 random (I.I.D.) sequences of 500 residues each, for all models in Rfam 9.1. Both programs were set to single-hit alignment. Due to the time requirements of trCYK, the simulations are somewhat shorter than necessary for complete convergence of lambda values under HMMER3.

when using un-structured models. This demonstrates that there has not been a major error in the adaptation of the probabilistic model of local alignments, since the performance in a sequence-only mode is similar to the previous results for sequence alignment.

With errors in adaptation of the HMMER3 local model eliminated as a source of variation, the next experiment was to consider the performance under consensus-only alignment, with no insertions or deletion permitted. Although an analytic proof of $\lambda = ln(2)$ is only available for ungapped primary sequence alignment, and not SCFGs, it seems likely that this principle also extends to the more complicated models, and I therefore expected to see much less variation in λ under this case. A custom version of trCYK was produced to run this, disallowing insertions and deletions and adjusting the transition parameters accordingly, and the results are given in figure 3.3. (The results for HMMER3 still use the standard model, including insertions and deletions.) As shown, ungapped alignments also produce λ estimates much closer to constant.

This suggests one of the assumptions of trCYK as a possible source for the variation in λ. Local parse trees, as defined in 2.3.1, are not unique, and there is generally more than one that will produce the same observed alignment. For instance, if a particular consensus position was in the missing data portion of the alignment, it would be impossible to distinguish whether a residue would be present in the complete sequence or whether it would be bypassed by a deletion state. The CM contains some information about which paths are preferred over others, but because there is no longer a one-to-one relationship between a particular alignment and a path through the model, an finding the best path is not equivalent to finding the best alignment. Figure 3.4 shows another possible example of indistinguishable paths, this time involving an insert state between two match-pair nodes. This ambiguity means that the reported score for an alignment does not capture all of the probability mass that would be assigned to paths with equivalent observed alignments. In the implementation of trCYK, we take the maximum likelihood of the possible paths which are consistent with the observed and aligned data. In a consensus-only ungapped model, however, there is only one possible path, eliminating these ambiguities. Similarly, an unstructured model is effectively already completely marginalized, and all missing data is outside the scope of the local parse tree (where it is already fully accounted for), and so these two tests both remove the effect of that assumption.

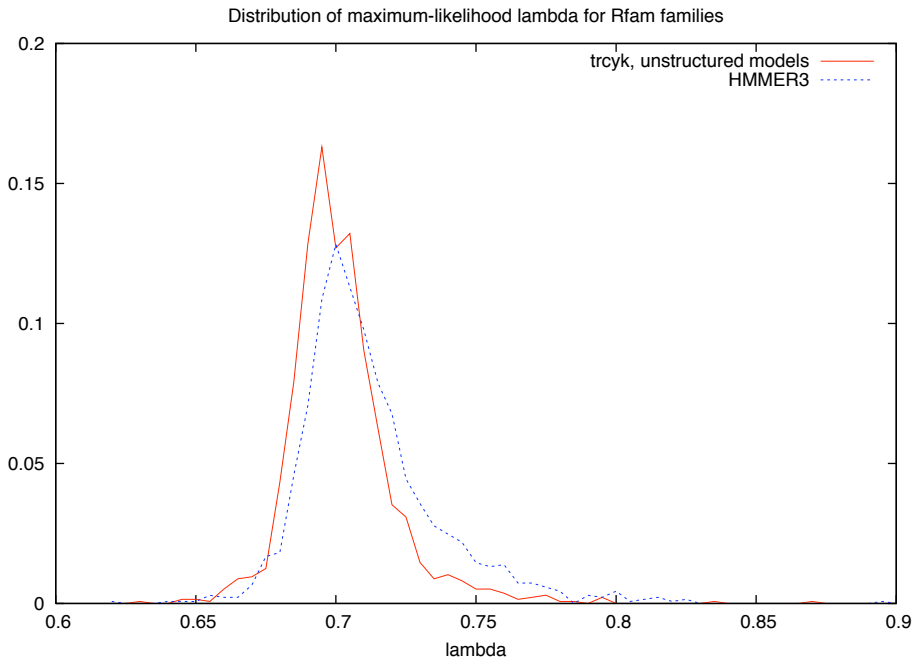


Figure 3.2: **Distribution of lambda for trCYK with unstructured models.** Maximum-likelihood fit of lambda for trCYK using models without secondary structure. Without secondary structure, CMs are effectively HMM-type models, and trCYK recapitulates the design of HMMER3, although there are differences due to having different priors and other details of model parameterization.

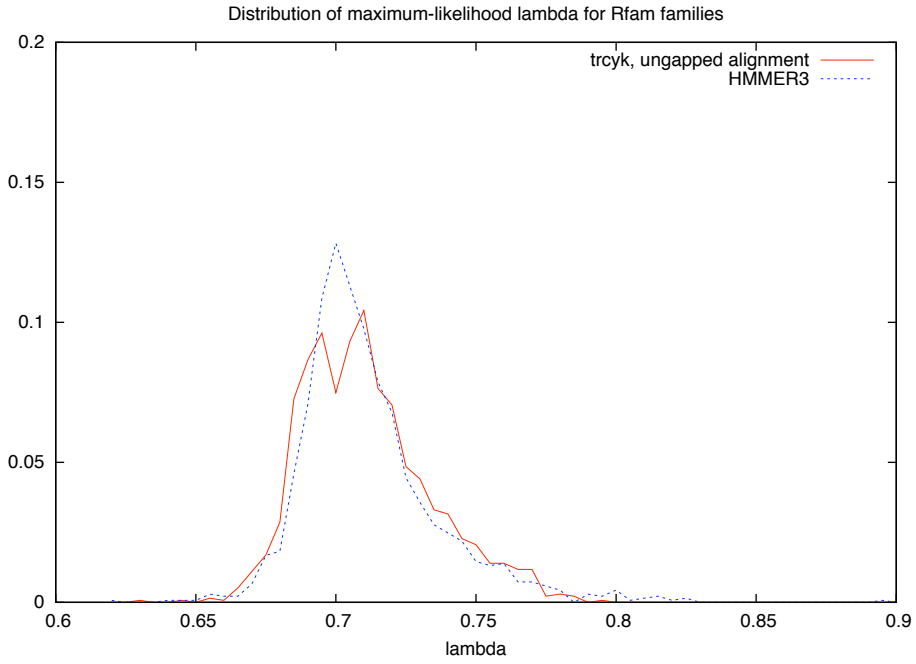


Figure 3.3: **Distribution of lambda for trCYK with ungapped alignment.** Maximum-likelihood fit of lambda for trCYK, modified to only consider consensus, ungapped paths through the model. Distribution of lambdas for HMMER3 remains as standard gapped alignment.

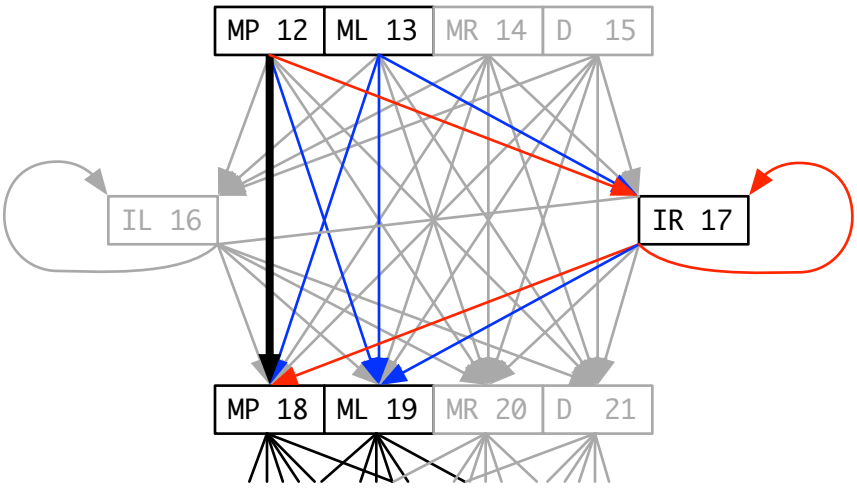


Figure 3.4: **Indistinguishable paths in alignments with missing data.** This detail view of a CM shows a small portion of an example alignment, where two adjacent residues are assigned to the 5' positions of two consensus pairs in the model, but the 3' residues are unknown. The heavy black line indicates the path expected to be the maximum likelihood path under normal model parameterization. Red arrows show a path which produces an identical observed alignment, since IR produces no residues when marginalized. Blue arrows show paths that use different states for the emission of the matched residues, but have the same effective alignment (these paths assert that one or both 3' residues are deleted, which is also consistent with the available data). Gray lines and states indicate paths that produce different observed alignments.

Ideally, we might instead integrate over all possible paths that produce the same observed alignment, however, this is technically quite difficult. To do so would require correlating multi-state paths through the profile, subverting the assumptions about number of children per state which is one of the key design features which reduces algorithm complexity from $O(N^6)$ to $O(N^4)$.

This hypothesis for the source of the variation in λ predicts two expected results. First, if a technically feasible method could be designed to integrate over the paths which result in the same observed alignment, then the distribution of maximum likelihood scores should exhibit a constant λ, likely with similar convergence behavior to that seem for HMMER3.

Second, total log likelihood scores summing over all alignments should also remove this observed variation, producing an exponential-tailed distribution equivalent to those seen for Forward scores, again with constant λ. This option is the more likely to be applied in practice; indeed, HMMER and Infernal have both already moved to total log likelihood scores (Forward and Inside, respectively) as their default scoring mechanism. I expect that this trend toward total log-likelihood scores will extend toward alternative alignment methods such as trCYK as well.

Chapter 4

Overview of parallelization in computer algorithms

In comparison with the previous material, the final results section, chapter 5, is somewhat more focused on details of implementation, and techniques for creating high-performance versions of existing algorithms. As such, it calls for some additional introductory material. For algorithms with high computational costs, including the dynamic programming alignment algorithms I have been discussing, implementations must be concerned with performance in order to be put into widespread use. There are typically two main routes to improved performance, which may be used in conjunction: heuristics which prune the solution space that must be checked, and software engineering to increase the raw speed of operations.

Parallel computing is an example of the latter, using computer hardware to perform many calculations simultaneously rather than sequentially. This chapter is presented as an introduction to the concepts of parallel computing; readers who are familiar with the subject may wish to skip forward to chapter 5, which concerns the application of parallelization to covariance model searches.

4.1 Classification of parallel systems

There are a wide variety of methods and approaches for parallel computing; some require specialized parallel hardware, while others use parallel software design and common communication networks. Here I introduce some of the core concepts from parallel computing, which can be used to categorize and compare different implementations of parallelism.

4.1.1 Control units, processing units, and data streams

In the influential description now known as the von Neumann model, a traditional sequential computer consists of two essential components: a central processing unit (CPU) and memory. The normal run of a program consists of many steps: the CPU loads the program from memory, and begins to follow the instructions it contains. The instructions allow many types of actions, including reading or writing data to memory, performing calculations on the data, and switching between different parts of the program instructions, depending on data values or potentially on user interaction. The run finishes when the CPU reaches an instruction that terminates the program.

The CPU in this model performs three main types of actions: control (determine which instruction to execute next), processing (arithmetic and logical calculations), and memory access (read and write). In modern sequential computers, and especially in parallel computers, it is not uncommon for the control and processing operations to be divided between different hardware units, and the control unit then provides direct instructions to the processing unit. Memory access either remains with the control unit, or in some architectures it may be shared between both units.

In the sequential computer we have a single control unit and a single processing unit (or a combined CPU). Because the processing unit operates on a single set of data at a time, this is classified as a single 'data stream', and so the computer is single instruction, single data, or SISD.

To achieve parallelism, we must have multiple processing units for simultaneous calculations, and we may also need multiple control units. If we have a single control unit, the processing units receive the same

instructions, so we can achieve a parallelism by providing different data sets to each processing unit. This arrangement is called single instruction, multiple data, or SIMD.

If instead we have as many control units as we have processing units, each pair of them can operate as an independent CPU, allowing a different flow of instructions depending on the data. In many cases, the control units may load the same set of program instructions, but because the execution is almost always conditional on the specific data, the instructions issued at any particular instant may be completely different between different control units. For this reason, this architecture is considered multiple instruction, multiple data (MIMD).

Although this classification scheme suggests the possibility of a fourth class – multiple instruction, single data (MISD) – such an organization has rarely been designed or used. If the different control units load the same set of instructions and the same data, they should produce the same computation, although this setup is sometimes used in systems requiring fault-tolerance or redundancy. (In those cases, for example some of the computer systems on the Space Shuttle, the processors will have different implementations or algorithms for a particular calculation, so as to hopefully avoid systematic errors.) If instead the control units load different sets of instructions, there is no concept of having the same data, because the instructions are what determines when and from what locations to read data from memory. There is some argument over whether pipelines, where data is passed directly from processing unit to processing unit, can be considered MISD. However, pipelines still usually only feature one overarching control unit which issues instructions sequentially, even if execution is staggered such that several operations are being conducted simultaneously.

The classification based on instruction streams and data streams which I have just described is known as 'Flynn's taxonomy', after its author Michael Flynn [80], and is illustrated in figure 4.1.

4.1.2 Memory organization and sharing data

A second way to differentiate classes of parallel systems is whether they have shared memory, and whether it is local memory, global memory or a combination of both. This classification is only really relevant when considering systems with multiple control units (i.e., MIMD); SIMD architectures with a single control unit

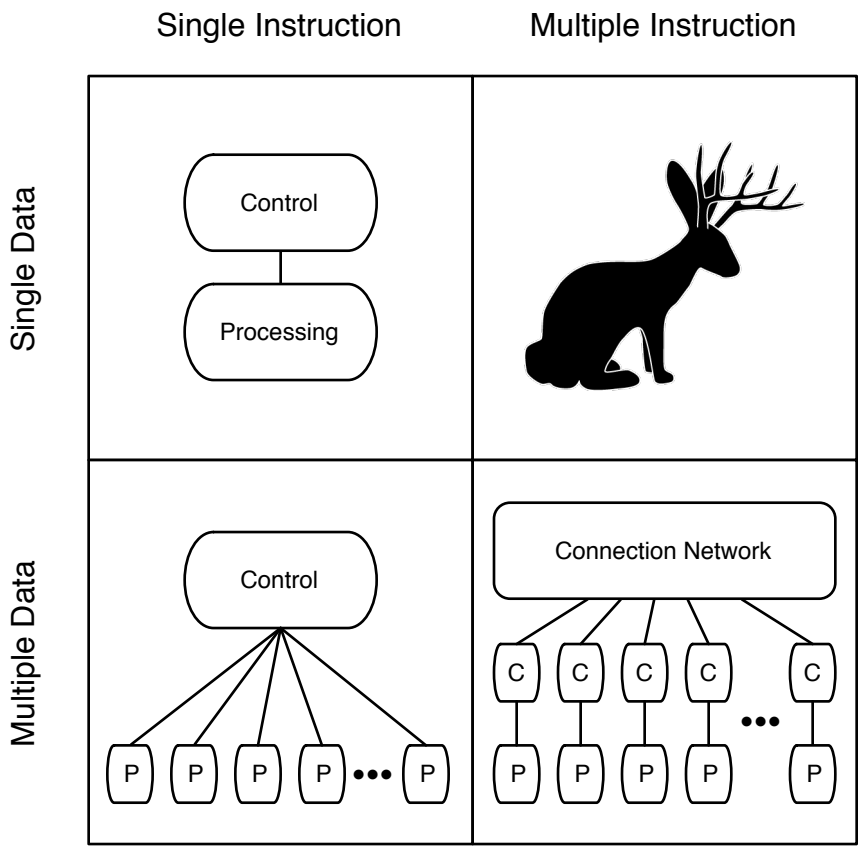


Figure 4.1: **Flynn's taxonomy of computers.** Possible computer architectures based on combinations of one or more processors operating on one or more data streams. Counter clockwise from upper-left, single instruction, single data stream corresponds to a serial (non-parallel) computer. Single instruction, multiple data corresponds to a single control unit, and multiple processors each operating on different data. Multiple instruction, multiple data features multiple control and processing pairs operating independently. Multiple instruction, single data is a mythical creature, with few confirmed sightings.

have direct and exclusive (non-shared) access to whatever memory units are on the machine.

Shared memory means that different processors have the ability to read and write the same memory locations. Although this is one way to communicate information between processors, care must be taken to ensure that conflicting changes are not attempted simultaneously, and that the data is not read while it is being changed by another part of the system.

Many architectures do not have shared memory at all; each processor has its own local memory which is not accessible by the other processors. These systems rely on message passing, where specific information is sent in packets across a communication network to the receiving processor, which must be programmed to receive it. The receiving processor then has control over where the data is written and what to do with it, rather than allowing the sender to write it directly to memory.

If a system does have shared memory, it may be either local, associated with a particular processor, or global and available uniformly to the entire system. Local memory is faster to access for the processor it is attached to, and slower for everyone else due to the delay imposed by the interconnection network, so local memory is considered 'non-uniform access' even when it is shared. Some machines with pure global memory have been built, but it is difficult to maintain bandwidth for many processors all requesting random access data, and most systems that use global memory also have local memory, which is used to store a copy of the program and any non-shared data.

Most modern parallel computers operate as a hybrid of local and shared memory: a relatively small number of processors share a uniform access global memory, operating as a mid-level unit; the cluster or supercomputer is composed of a large number of these nodes, and memory is not shared between them, so network communication is also required.

4.1.3 Synchronization

Synchronization refers to how often and how much information is exchanged between different processing units. Unlike the previously mentioned classifications, it has no discrete boundaries, but operates in a continuum between complete synchronization and completely independent processes which do not synchronize

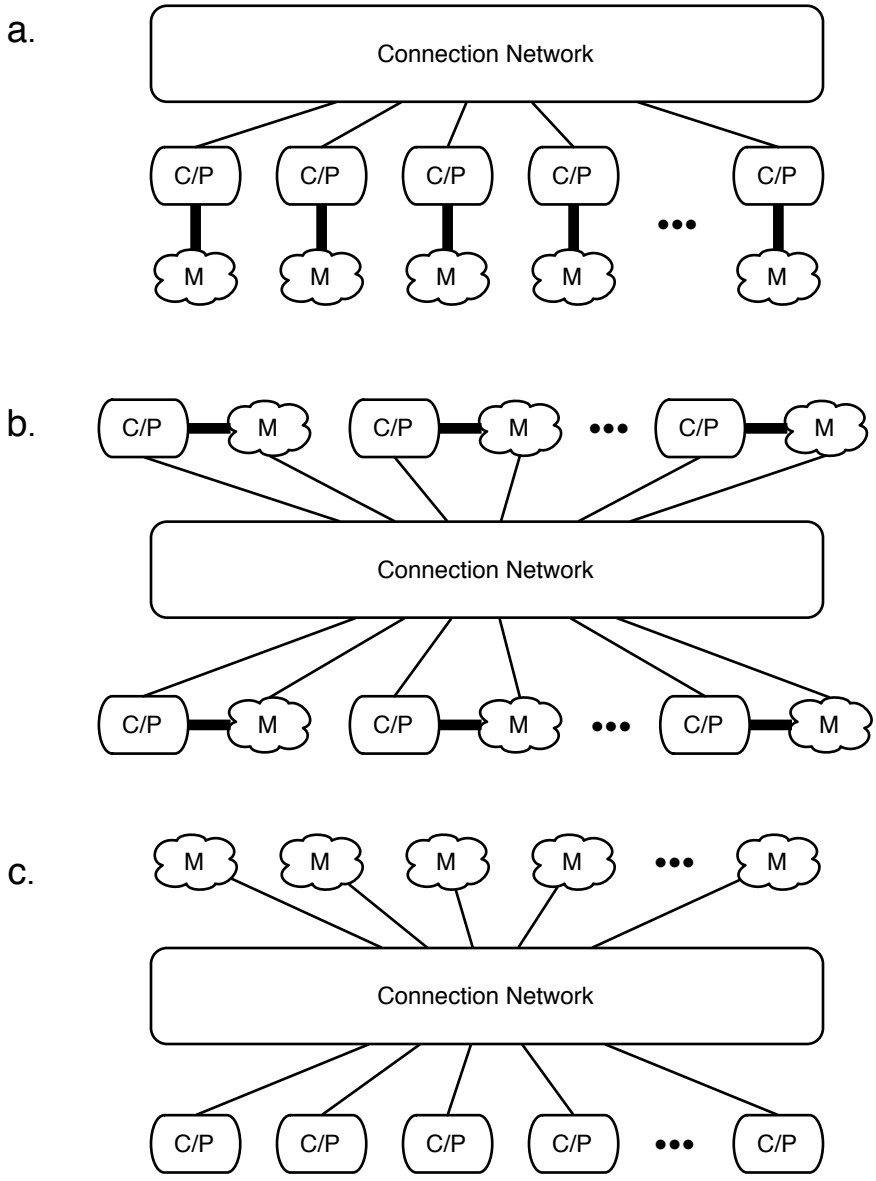


Figure 4.2: **Private, local, and global memory configurations.** Private memory is used in message-passing systems (a). If memory is shared, it may have preferential access for its local processor (b), or it may be unassociated with a processor and uniformly accessible globally (c). Most systems with global memory also have local memory, operating as a hybrid of (a) and (c) or (b) and (c). C/P indicates a control and processing pair, M indicates a memory unit. Thicker lines show connections which are generally faster/higher bandwidth.

at all. To a certain extent, the level of synchronization is determined by the architecture and memory organization – shared memory designs are typically more highly synchronized than message-passing systems, and SIMD processors are completely synchronized, with a new instruction being broadcast every clock cycle.

4.1.4 Interconnection networks

Interconnection networks describe the communication pathways between processors, either physically (what communication routes are available) or in algorithm design (which units actually communicate in a particular program). The topology of connections between processors is an important topic in the classical design of parallel computers, but it is of little relevance to the type of SIMD architecture I will be concentrating on, and thus I will not discuss it further. That said, interconnection networks are particularly important when designing algorithms for heterogeneous systems, where a network of primary processors may each have an exclusive connection to additional resources such as co-processors.

4.2 Granularity, and common parallelization strategies

Conceptually, the main strategy of parallelization is to split a given problem into many smaller sub-problems which can be solved (relatively) independently, and to send each of them to a different hardware unit. The scale, or 'granularity', of those sub-problems can vary greatly, depending on the class of computer resources that are being targeted.

4.2.1 Multiple instances on multiple computers

One of the simplest and most familiar types of parallelization is when multiple independent tasks are run on a computer cluster. For example, if several sequences are to be searched against the same database, each computer can be assigned to search a single sequence against the database. Each computer has its own running copy, or 'instance' of the search program, and they start, output results, and finish independently of each other. Because this approach requires accessory programs or user intervention to both start the individual

tasks, determine when all have completed, and process the disparate results, this approach has essentially no synchronization. Problems that can be parallelized in this way are said to be 'embarrassingly parallel'.

4.2.2 Single instance on multiple computers with message-passing

In the above example, it would generally be up to the user to organize and coordinate the results from multiple runs of the program. When a program is expected to run across a cluster routinely, or when the sub-problems are somewhat less easily split, it is advantageous to have a more centralized and automated approach. In an alternative approach, one program can spread itself out over multiple processors, automatically dividing the problem as necessary. Frequently, one processor will be designated the 'master', scheduling sub-problems for the other processors as they become available, and distributing information and collecting results over whatever communication network connects the machines. In addition to generally requiring less active management from the user, this type of parallelization can reduce the overall computational cost, by avoiding unnecessary data transfer. Again using the example of searching sequences against a database, each processor could be assigned a particular slice of the database. Then, the query sequences could be sent to each processor, covering the database without any single processor needing to have the entire database available. This class of approach is generally implemented as a message-passing system, because the processors have relatively modest needs to communicate with each other. By far the most commonly used message-passing libraries are various implementations of the Message Passing Interface (MPI) standard [32, 83, 101, 110], although there have been others [238].

4.2.3 Threads: multiple control streams on a single computer

At a somewhat finer level of granularity, each computer need not be restricted to running a single control stream. 'Threads' can share execution time on a single machine. On a machine with a single processing unit, this might mean switching between tasks while one is waiting on a slow operation (such as reading data from a hard drive), but more typically, threading is used in conjunction with multiple processing units, or 'cores'. Although some approaches to threading still operate under a message-passage paradigm, threads

have the alternative of shared hardware memory within a given node. With a shared memory model, a thread can communicate by changing values that the other threads can read. There are myriad implementations of different approaches to threading, often as extension libraries to existing languages such as POSIX threads and OpenMP. Alternatively, there are also inherently parallel programming languages which have been explicitly designed with the concepts of parallelism in mind, including Erlang and occam. Either local message passing or shared memory has a much shorter timescale than network communication, and as a consequence, threaded programs communicate much more frequently, and are more tightly synchronized.

4.2.4 Parallelization on a single processor (SIMD)

Even at the finest granularity and most synchronization discussed so far, each hardware unit will be following its own set of instructions, i.e., they are MIMD systems. However, in cases where the algorithm to be performed is highly stereotyped, with the same operations being performed many, many times over different data values, it is possible to synchronize further still. Current processors have capabilities for what is known as 'vector operations', originally designed for multimedia and gaming applications, but usable for general purpose computing. Vector operations parallelize by issuing a single instruction at a time, which is executed simultaneously on a row, or vector, of data values. This allows SIMD calculations, a very local level of parallelization, working within a single processor. The number of values which can be calculated simultaneously varies; for a given hardware platform, vectors will have a fixed number of bits, with a trade-off between the number of bits allotted for each data unit (affecting the precision and the range of the values it can take) and the number of individual values that can be packed in the vector. Vector operations are accessed by means of code libraries that reflect the capabilities of different hardware platforms, such as SSE for Intel family chips, and Altivec/VMX for the Power architecture.

4.2.5 Combining multiple levels of parallelization

The levels of parallelization generalized above are not mutually exclusive; several of them can be used in a nested fashion. For instance, a program distributed over a cluster using message-passing could also be using

vector code for finer-grained parallelization on each of its nodes.

4.3 SIMD Parallelization of Smith-Waterman

The work in chapter 5 concentrates specifically on SIMD parallelism for dynamic programming alignment algorithms. By way of example, I wish to take a brief aside to discuss the history of SIMD parallel implementations of the classic Smith-Waterman alignment algorithm. The approaches taken, and the levels of performance increase associated with them, should be illustrative for other dynamic programming alignment methods, which are fundamentally similar.

As in the general discussion above, Smith-Waterman has been optimized at many scopes, from distributed multi-processor versions to SIMD approaches, and also has been implemented using algorithm-specific hardware in the form of field-programmable gate arrays [105, 162, 266]. I am limiting this overview to SIMD approaches on general-purpose computers.

SIMD parallelization is sensitive to various design choices, especially when applied to computationally intensive algorithms. Some of the key advances in parallel implementation of Smith-Waterman have reflected underlying changes in the choice of exactly which calculations will be performed simultaneously, and how the data will be arranged to support that. These changes reflect the need to simplify the inner-most loop of the algorithm, which is executed vastly more often than the remainder of the code. Specifically, two events must be avoided whenever possible: first, dependency between values, such that an operation must wait for the results of a previous calculation; and secondly, conditional branching of the code, as misprediction of the branch also results in stalling while the correct branch is loaded.

Although an early implementation parallelized across multiple alignments, having each element in the vector correspond to a different sequence from the database [1], most approaches have sought to parallelize the alignment of a single pair of target and query sequences. Considering the rectangular matrix of values that must be calculated to determine the alignment, the vector may include elements from the same row, the same column, or a diagonal. (Parallelizing either in rows or columns may be considered identical for this

problem, merely by switching which sequence is the query and which the target.) The approach of Wozniak parallelized across the minor diagonal, since that minimizes the amount of dependency within a single vector [265]. Each matrix cell depends on the values to the left, above, and on the upper-left diagonal. Thus, a set of cells in a lower-left to upper-right diagonal do not depend on one another. This implementation used 4-fold parallelism, and claimed roughly 2-fold speed-up over an equivalent scalar version.

Later implementations have sacrificed that degree of independence, finding a substantial trade-off in the speed of loading values for a column instead, and tried to minimize the effect of the vertical dependency by only evaluating that path when it could possibly contribute to the score [208]. (Due to the nature of local alignment, the alignment score must exceed the score of a single-residue gap before one even needs to consider adding a gap, and this is untrue for many positions in the matrix.) Because a column-wise vector corresponds to a single residue in the database sequence, this also enables the creation of a query profile to increase the speed of looking up the weights of matches. Rather than indexing the substitution scores by query residue and target residue, the query can be pre-processed at the beginning of the search to produce a matrix indexed by sequence position and target residue. Thus, the substitution scores are also contained within a column-wise vector that can exactly correspond to the matrix values being calculated [208]. This approach improved to approximately 6-fold speed-up, although it did simultaneously move to 8-fold parallelism.

A more recent parallel adaptation of Smith-Waterman also uses vectors parallel to the query sequence (i.e., vertical columns as described here). However, rather than processing cells in the order which they naturally occur, Farrar re-arranges them into a striped pattern such that each vector contains values from distant segments of the sequence, and normally adjacent cells are in subsequent vectors [77]. This arrangement minimizes the number of shifts to be performed when considering the diagonal dependency, which is misaligned in most other layouts (shifts generally being slow compared to arithmetic or logical operations). Furthermore, this arrangement moves the vertical dependency from within a single vector to between adjacent vectors, moving the conditional check on that path from within the innermost loop to just outside it. With increasing hardware capabilities, this most recent version again increases the degree of parallelism to 16-fold, but shows approximately 7-fold and 2- to 5-fold speed improvements over re-implemented versions

of the Wozniak and Rognes algorithms, respectively (no comparison to a non-parallel version was given).

Chapter 5

Parallelization and Fast Filtering for CMs

5.1 Abstract

Covariance models (CMs) are powerful tools for identifying and aligning structural RNA genes. However, the algorithms dealing with CMs have high computational demands, which has limited their practical application. Often, structural RNA searches will first employ a primary sequence based filter, which cannot capture the long range pairing interactions of RNAs. Here I present a multi-stage filtering strategy which incorporates pairing information at all stages. This approach identifies a significant number of hits which are missed by other filtering methods, and combining the results of both sets of filters provides better sensitivity than either method alone.

5.2 Introduction

Covariance models (CMs) are probabilistic profile models for structural RNAs and their secondary structures. Including information about base pairing and covariation allows CMs to significantly out-perform sequence-

only search and alignment methods when looking for RNA genes [82, 185, 187]. Unfortunately, though, CMs have historically also been among the slowest and most computational intensive alignment methods, particularly compared to sequence-only approaches, but even compared to more heuristic models including secondary structure.

CM search and alignment are conducted by dynamic programming (DP) algorithms and they are efficient in a technical sense: they guarantee an optimal solution without brute force examination of all possibilities, and the required time is a polynomial (rather than exponential) function of the size of the inputs. A fully general SCFG has memory complexity $O(N^3)$ and time complexity $O(N^6)$; this means that in the worst case scenario, doubling the length of the query requires 8 times as much memory, and 64 times as much time to complete. CMs simplify this somewhat by their profile structure, which limits the number of transitions per state and the number of branching structures, reducing the time complexity to $O(N^4)$. Memory requirements remain at $O(N^3)$, although divide and conquer approaches can reduce this to $O(N^2 \log N)$, at the expense of a constant-factor increase in running time [66]. This still compares quite poorly with sequence-only alignment methods, which are generally $O(N^2)$ in both memory and time.

Asymptotic scaling behavior is not a complete description of the necessary running time for a program – a coefficient represents the minimal unit of computation, and an $O(N^4)$ algorithm with a very small constant could be faster than an $O(N^2)$ algorithm with a very large constant, at least for small problem sizes. However, this is not the case when comparing CMs to their primary sequence counterparts, and the computational requirements are burdensome for many problems of interest. The benchmark I will be using for evaluation later in this work involves searching against a 10 Mb database, and even at that amount of sequence only the smallest models can be searched in under 1 CPU hour, with the average being more than 12 CPU hours each. The problem is only magnified when searching against a larger database, such as a large eukaryotic genome, or larger models (the largest model considered, a domain of rRNA took 72 CPU hours, and is still much smaller than complete rRNA models).

5.2.1 Improving CM speed by filtering and search space reduction

Because of the computational requirements for CM search, many different approaches have been proposed and used for increasing the speed of search and alignment. Broadly speaking, these approaches fall into two categories: those that attempt to increase the speed of the underlying algorithm, and those that provide a filter that is applied before the final alignment stage. The distinction between these two groups is not solid, though, as several of the techniques that will be described below may be applied in either style.

Historically, the most frequently-used filtering strategy has been to apply a primary-sequence-alignment-based cutoff to the search before applying the slower CM search. For example, Rfam uses a BLAST search to bring computation time down to a feasible level when preparing a new database release [85]. Although these filters remain necessary for a project on the scale of Rfam (which even with them requires several CPU-years), they are troubling and represent a potential loss in sensitivity, precisely because they ignore the secondary structure which gives CMs an advantage over primary-sequence-based search for non-coding RNAs.

Previous experience has demonstrated that more complex filtering strategies can be highly effective, providing both better sensitivity and sufficient speed-up. The tRNAscan-SE program uses a combination of two tRNA detection algorithms as filters to a final CM-based alignment stage. This combination is highly effective, resulting in a tool which is deployed at genome scale without the need for additional filtering steps. Although this approach showed that there clearly exist filtering strategies with better sensitivity/speed trade-offs than BLAST, the design of the filters for tRNAscan-SE is highly model-specific, and cannot easily be extended to generic models of RNA. Thus, much work has concentrated on mechanisms to automatically produce appropriate filters given a CM, rather than requiring human intervention and expert knowledge.

Much recent effort in improving the speed of CM search has gone into the design of HMM-based filters. Although HMMs, like BLAST, are based on primary sequence and do not consider secondary structure, they have several advantages. First, profile-based searches have been shown to have superior sensitivity compared to single-sequence comparisons [195], and they can reduce the overall number of searches required by grouping sequences into families rather than performing one-to-one comparison for all combinations. In addition, the automatic training of HMMs allows parameters to be more easily adjusted on a per-family basis,

and can even be used to determine whether a particular style of filter is likely to be appropriate for a given family [184].

Weinberg and Ruzzo produced a series of HMM filtering techniques built directly on top of the Infernal package; the first of these was rigorous filters [255]. The rigorous filters used a very direct mapping of CM to HMM states, cleverly parameterized in such a way that the score of any sequence under the HMM was guaranteed to be greater than or equal to the score of the same sequence under the CM. This meant that any given bitscore threshold for the CM could be directly applied to the HMM, and all sequences that would pass the CM would also pass the HMM, guaranteeing 100% relative sensitivity. These filters were generally effective, but quite variable in performance between families, and they could not easily be tuned for a trade-off between speed and relative sensitivity.

Subsequent work introduced the Maximum-Likelihood heuristic filters (ML-heuristic) [256]. Rather than forcing the CM score to be a lower bound on the HMM score, the ML-heuristic produces an HMM which is maximally similar to the CM, i.e., it attempts to optimize the correlation in scores between the HMM and the CM. Rather than training an HMM on the original multiple sequence alignment (MSA), it trains HMM parameters on the MSA produced by running the CM as a generative sequence model. The HMM filters used in Infernal 1.0 are a reimplementation of the ML heuristic filters, and use sampling during the required model calibration step to determine an HMM threshold providing 99.3% relative sensitivity or 2% of the database, whichever is more permissive [184, 186, 187]. This simulation also provides time estimates for the search both with and without filters, enabling automatic shutoff of the filter if the predicted speed-up is small.

The same HMM-based proxy models can also be used for direct alignment speed-up, rather than pass/fail filtering. In this case, a preliminary HMM alignment is used to determine portions of the alignment space that can be excluded as being extremely unlikely to contain the true alignment. These boundaries, or bands, can then be transformed into the SCFG alignment space and limit the number of possible alignments to be calculated. This approach was used in RNACAD [27], and later re-implemented in Infernal [184, 187].

The filtering and banding techniques described so far are strictly sequence based, ignoring secondary structure. The same general approaches can be applied with techniques that include some structural elements,

although by necessity these models must be simpler than full CM alignment in order to offer performance gains.

Weinberg and Ruzzo have also described two methods for including limited structural information in an HMM-filter. The sub-CM method embeds a small CM inside a larger HMM for a hybrid semi-HMM model [254]. The residues covered by the embedded CM are scored under the more complex CM structural rules, while the majority of residues are scored under the more efficient HMM. Because all residues inside the CM portion are subject to the more complex processing, this method is limited to small segments and short-range interactions, such as a terminal stem-loop.

The store-pair technique accomplishes a similar objective, but it uses HMM-type states exclusively by enumerating the possible combinations for selected base-pairs by branching [254]. Consider a particularly important but low-identity base-pair: an HMM can model only one residue at a time, so rather than having a single state which emits all of the possible bases, create a branching structure with a separate state for each of the bases. Each of these branches then has identical copies of the states for the intervening residues, up to the residue which closes the base pair, which is scored under the conditional probabilities given the opening base. Once the base-pair is completed, the branches can re-merge to continue the normal linear profile structure. This method trades off overall number of states (and therefore computation time) for increased sensitivity; the increase in states depends on the distance between the residues of the base pair and whether two or more stored pairs are nested inside each other, in which case it is subject to a combinatorial explosion.

Other structural filters have used more direct specifications about the structural elements of the ncRNA. FastR looks for elementary units that are stacks of canonical or wobble base pairs (possibly with a limited number of insertions or deletions), and assembles them into nested or multi-loop structures, with length constraints on the elements and the distances between them [14]. These filters end up being similar to those found in structure description languages such as RNAMOT, except that these filters eschew all primary sequence information, and software is provided to produce filters automatically from an input ncRNA sequence [273]. The drawbacks to this and similar approaches is the rigidity of their rules; a true positive can easily be lost through non-canonical base-pairs, stems too short to meet the recognition threshold, and indels large enough

to throw off the length constraints.

Query-dependent banding (QDB) is a banding technique which limits the DP matrix based on subsequence lengths. Consider a base pair in a non-coding RNA: those two positions specify a length of sequence which they enclose, and for each state in a given model, the distribution of those lengths will be characteristic of that family. Thus, rather than considering the alignment of all subsequences up to some window length W, we can limit our search to those subsequences that have a length likely according to our current position in the model. Because these bands depend only on the query model, they can be efficiently pre-computed [185]. In the most recent version of Infernal, QDB with Viterbi (maximum likelihood) alignment is also converted to be used as a filter, enabling the use of the more complex Inside calculation in the final alignment stage [184, 187].

5.2.2 Improving general DP speed by parallelization

A few examples from the history of SIMD parallelization of sequence alignment (specifically Smith-Waterman alignment) were mentioned in section 4.3; the same ideas and methods have also been used much more broadly across sequence and profile alignment. The profile HMM packages HMMER and SAM have both used SIMD parallelization [69, 95]; the most recent release of HMMER includes a new SIMD implementation including the efficient memory layout techniques use by Farrar [69, 77].

Most of the previous applications of parallelism to SCFG algorithms have been at the somewhat coarser-grained MIMD level. RSEARCH and Infernal both have MPI implementations for distribution across a cluster [140, 184]; these versions parallelize by splitting the sequences to be searched across different processors. Liu and Schmidt presented an alternative method of parallel CYK alignment, splitting each alignment into many blocks that can be computed simultaneously as a wavefront [165]. Massive MIMD parallelism approaches have been applied to RNA folding using genetic algorithms [222, 223].

SIMD parallelization for SCFGs and non-coding RNA in general has received less attention. Both of the earliest applications of SCFG models for RNA secondary structure prediction were implemented for massively parallel MASPAR machines [74, 94, 96]. In the approach of Ellingworth and Eddy, each row

of the processor grid was assigned a different sequence to align, although few additional details about the parallelization of the algorithm were given. There is a very recent implementation of an FPGA accelerator for CYK alignment, but it is unfortunately based on the now quite outdated Infernal 0.55 technology [58]. Krishnan et. al. have attempted to parallelize the more complex problem of RNA structure prediction with pseudoknots, producing an implementation of PKNOTS [204] for the Cell processor, but with dismal results – the parallel implementation generally taking longer to run than its scalar equivalent [147].

5.3 Approach

I employ two complementary strategies to reduce the computation time of CM search. First, parallel computing allows the calculation of multiple values simultaneously. In addition, a simplified alignment model reduces the overall number of operations which must be performed. These strategies will be complined in a multi-stage pipeline which moves from simple models and high parallelism at the beginning, to full models and moderate parallelism at the end.

There are several different scopes for parallelism - here I am targeting very fine grained, tightly integrated Single Instruction, Multiple Data (SIMD) vector operations. Higher-level cluster parallelization with MPI is already available for Infernal; although I will not directly address it here, these two levels of parallelization are amenable to integration, and will likely be used in a combined approach at a later date.

5.3.1 Pipeline design overview

The overall design of the filtering strategy closely mirrors that used in the HMMER3 pipeline [69]. The first stage uses an ungapped model of alignment, producing short consensus hits, and operates in very limited numerical precision. By representing scores as 8-bit unsigned short integers (range $0..255$, mapped to -60 to 20 bits), it is possible on most hardware to calculate 16 values concurrently. Because the score range is very limited, most true sequences will reach the score ceiling (saturate); however, this is sufficient to indicate that a sequence has passed the filter and should proceed to the next stage. The unique design of the filter to

produce ungapped but structured consensus hits will be discussed in more detail later.

The second filter is much closer to standard CYK alignment, re-incorporating insertions and deletions. It is implemented in medium-precision with 8-way parallelization, using signed 16-bit integers, each representing $1/500$ bit units for an effective range of -65 to 65. Although this level of precision gives good resolution of scores, it is not sufficient as a final step because rounding errors may accumulate, and some high-scoring sequences will still saturate the score range.

The final stage is identical to ordinary scalar alignment, using 32-bit floating-point numbers, except that it is implemented with 4x parallelization. This step may also provide for the return of a CYK alignment trace, rather than just the score of the sequence as in earlier steps.

5.3.2 Designing a structural filter

The design of the first-stage filter asks a crucial question which I wish to examine further: what does an ungapped consensus hit to a CM look like? The intuition of what ungapped local alignment means in a primary sequence context is a set of contiguous positions in both the query and the target, for which the sequences are identical or very similar, and of the same length. This concept has been used extensively as a seeding mechanism for sequence alignment: the tuples of FASTA [196], the word-hits of BLAST [6, 7], or even the MSV stage of HMMER3 [69]. Extending this idea to the branching tree structure of a CM introduces certain questions. It is not obvious *a priori* whether, for example, such hits should incorporate only paired elements, only unpaired bases, or a mixture of both. If a hit includes a base pair, we could require it to extend such that all the sequence between the the two bases is also included in the hit, but that could very well be too stringent.

The design of the multi-segment CYK (MSCYK) filter was based on the following principles. First, a hit should correspond to an unbranched range of CM states, which may contain single-stranded residues, base pairs, or both. As a corollary, hits should be scored according to the probabilities in a particular model segment, rather than being judged against deterministic criteria such as, for example, “four canonical base pairs”. Secondly, hits may be assembled together to produce arbitrary nested and multiloop structures, with

intervening un-aligned sequence and not necessarily maintaining consistency with the original CM. This criteria is not because we expect real hits to be significantly re-arranged relative to the consensus, but because it is much simpler to find all sets of hits rather than only sets of hits that are consistent with the underlying structure. In practice, high-scoring hits that are rearranged relative to the model are expected to be rare. Figure 5.1 shows examples of the types of hits which will be found by this design.

These design goals led to the creation of a very simple grammar, which is used to encapsulate the CM and provide the connections for the nested and sequential hits. This grammar has two non-terminals: M, which produces a model segment, and S, which produces unaligned sequence and connects model segments under the following rules:

$$\mathrm{S} \rightarrow \mathrm{S}a|\mathrm{SM}|\epsilon \qquad\qquad \mathrm{M} \rightarrow x_{w..z}|x_{w..y}\mathrm{S}x'_{w..y}$$

where $w..y$ or $w..z$ is a range of CM model states, and x and x' are the corresponding 5' and 3' residues respectively. The ending state for the two emissions of M may either be a model end (z), in which case the residues produced are continuous, or it may be a non-end state (y), where the 5' and 3' residues are emitted in separate segments, and the intervening region recurses to the S non-terminal. The cases are similar in nature to the wedge-type and v-type subproblems described in section 2.8.3. Although this grammar has very few free parameters with which to control the types of sequences it prefers, the small size presents an advantage for speed since, in general, each non-terminal represents additional memory requirements and each additional transition is another comparison operation to be included. Furthermore, since the model segments are by definition unbranched, there is only one state for which a bifurcation must be calculated, which is important as bifurcations require a disproportionate amount of time compared to other types of states. The grammar is known to be ambiguous – there may be more than one parse which still assigns all residues to the same states – but in this low-resolution classification, the loss of probability mass from the most likely parse is probably acceptable.

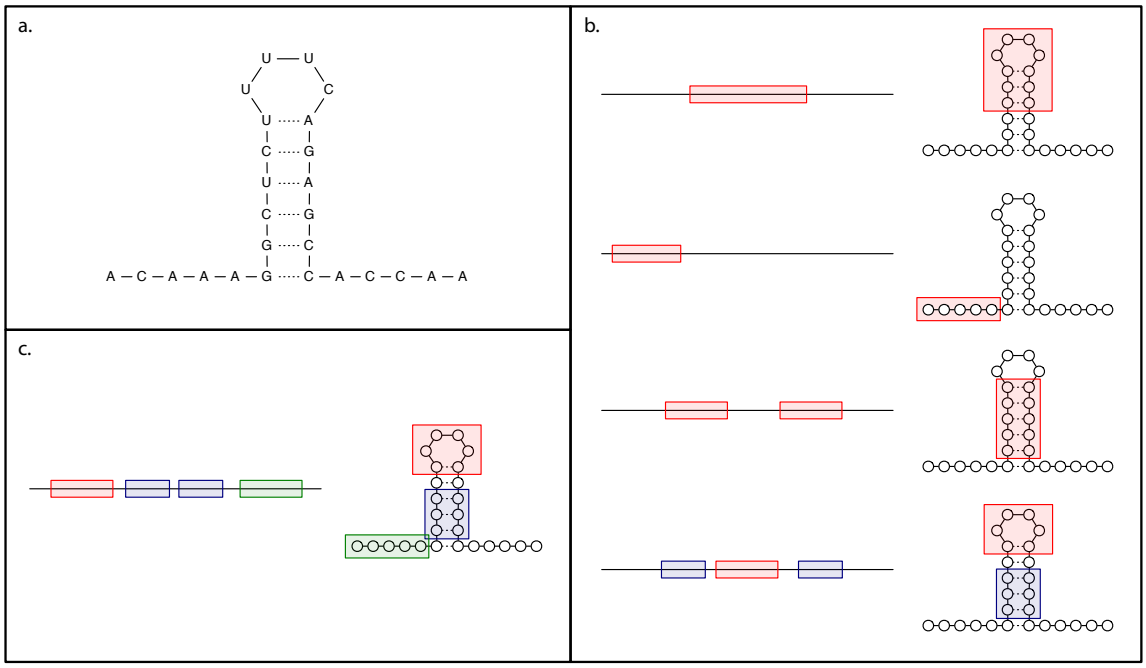


Figure 5.1: **Examples of possible consensus hits. a.** The structural model for the histone 3 RNA hairpin. This family has a 6 base-pair stem with a tetraloop, and 5 unpaired bases on each side. The sequence given is the maximum likelihood sequence under the CM. **b.** Examples of typical consensus hits to the model. For each of the examples, a simplified representation of the model is at right, and the the horizontal line represents a sequence being searched. The colored boxes correspond to a particular ungapped segment. If there is more than one such segment, they are given different colors to distinguish them. From top to bottom, a match to the hairpin, to an unpaired region, to just the stem portion (note that the two sequence portions are separated by an arbitrary distance), and finally a nested pair of matches to the stem and loop regions. This nesting is consistent with the parent structure, and so is representative of what we might expect from real examples of the sequence. **c.** Example of an atypical hit to the model. This series of matches each has a consensus match to a portion of the model, but their arrangement is not consistent with the original model. This type of hit is possible in the first stage MSCYK filter, but expected to be rare, and will not give a good alignment in the later pipeline stages.

5.4 Methods

5.4.1 Hardware architecture

SIMD code is by necessity closely tied to a particular hardware platform. In the implementation presented here I have targeted the x86 chip family for its ubiquity, using the SSE and SSE2 instruction sets, present on almost all Intel processors since 2000 and all AMD processors since 2003. This algorithm uses only a small subset of SSE instructions, which are commonly available on alternative platforms, and the algorithm design itself should be fairly portable to other SIMD architectures.

5.4.2 Memory organization

One of the key factors governing the efficiency of a SIMD program is the organization of memory; in order for individual elements to be processed in groups as vectors, they must be physically adjacent, and aligned to the higher-level boundaries imposed by the architecture. Because the units in a dynamic programming matrix are typically represented as large arrays or matrices of data even in a scalar implementation, this does not at first glance seem to present a particular difficulty. However, in practice, it is not necessarily obvious how best to segment the DP array into vector units.

The earliest approaches assigned each element of the vector to the same coordinates of different sequences [1], but it is difficult to maintain efficiency in this paradigm for two reasons. First, the sequences must be nearly identical in length or efficiency drops as the ends of shorter sequences are reached but computation continues for the longest sequence. Second and more importantly, because multiple alignments are being performed at once, the memory for each alignment must be allocated and available simultaneously, significantly increasing the memory footprint.

For this reason, the standard is instead to parallelize within a single alignment. In primary sequence alignment (either sequence-to-sequence, such as Smith-Waterman, or sequence-to-model, such as HMMs), the dynamic programming matrix may be thought of as a rectangular array, with the query corresponding to one axis and the target to the other, and the optimal alignment being a complete or partial path through the

lattice (depending on the type of alignment). The individual elements, or cells, may be grouped into vectors by choosing a set from a row, a column, or a diagonal. The minor diagonal (lower left to upper right) features the least amount of dependency between adjacent cells, since none of them depend on each other [265].

There is a trade-off, though, as assembling (for example) residue scores for a diagonal vector is time-consuming: the elements do not share a common position in either sequence, and so each element must be looked up independently and sequentially. More recent approaches have sacrificed the low dependency of diagonal vectors for faster look-up of scores with row- or column-based vectors [77, 208].

Even then, there are multiple ways to organize the cells into vectors. The most obvious is to simply group elements sequentially. In a four-element vector, this would mean that the first four elements are grouped into the first vector, the second four elements in to the second vector, and so on. This is intuitive, however it also means that each element within the vector is directly dependent on the element adjacent to it in the vector. An alternative approach was demonstrated by Farrar [77]. Rather than being split sequentially, the vectors are assembled in an interleaved or 'striped' fashion (see figure 5.2). The first vector in this case would contain the first element of a row (or column), then the element 25% of the way through the sequence, the element halfway through the sequence, and finally the element 75% of the way through the sequence. In the second vector, each cell is offset by one from the corresponding cell in the first vector. In this way, the direct dependency is moved from the previous *cell* to the previous *vector*. Although more complex to understand, this layout has significant speed advantages [77].

CM alignments (and other SCFG-based DP problems) should benefit from the same layout considerations that have been used in primary sequence alignment. This is complicated somewhat by the additional dimension of the DP matrix: rather than a rectangular 2D matrix, CMs have a 3D matrix with coordinates usually represented as v (model state), j (subsequence end position), and d (subsequence length). At any given point, though, we may simplify this by considering only a particular slice which varies with j and d. This is similar to a 2D matrix for sequence alignment, except that not all combinations of j and d represent valid positions: a subsequence which is longer than its distance from the start of the sequence has no meaning, so we need not consider any cells where d is larger than j. If d is allowed increase to the full sequence length L, then this

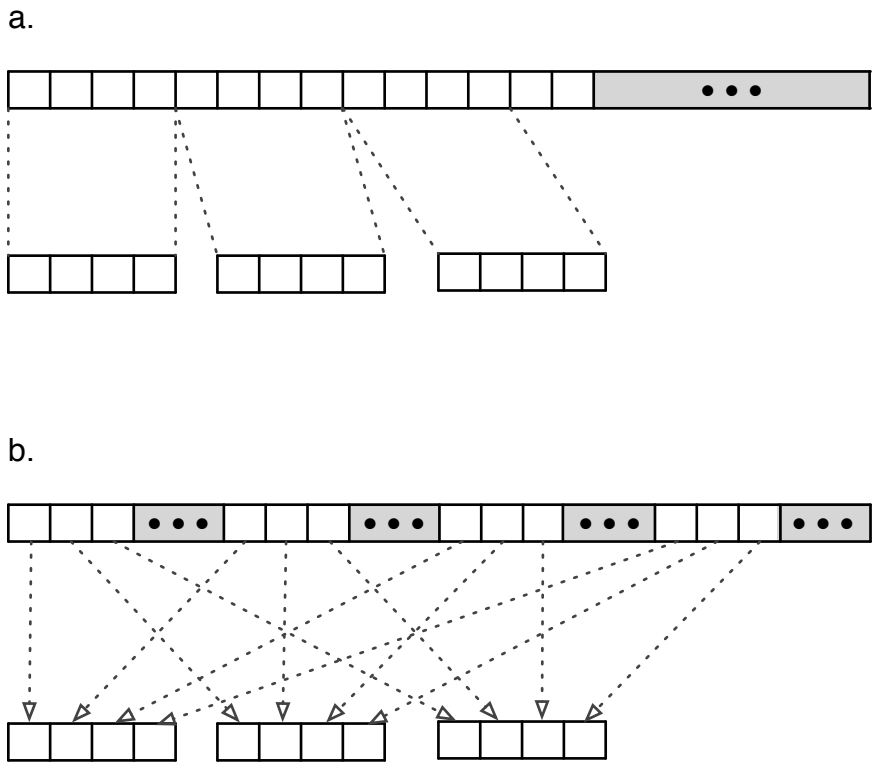


Figure 5.2: **Sequential and interleaved memory layouts. a.** Sequential segmentation of a row of values; each element is grouped with its naturally adjacent positions. **b.** Interleaved or striped memory layout. Elements are grouped from disparate portions of the natural row order, and 'adjacent' elements are now found in subsequent vector units.

slice of the matrix is triangular in shape. This is usually true in alignment contexts, where a putative homolog has been identified, and the total sequence is not much larger than the expected length of the model. The other main context is a scanning context, where the range of j is much larger than the expected size of the RNA, such as when searching a genome. In these cases, the range of d is usually limited by a window size parameter W. Although there are a few invalid cells at the beginning of the sequence where $j < \mathrm{W}$, most of the rectangular L by W matrix is filled in (see figure 5.3).

The move from scalar to vector calculations almost always involves some increase in the number of matrix cells which must be calculated, even if they correspond to impossible alignments. Unless the size of the matrix can be divided evenly into the vector size, the last vector in a row will have a few empty or placeholder cells which we cannot avoid calculating. For the triangular matrix of CM alignment, this means that the stair-step of the diagonal will increase in units the size of the vector, rather than units the size of a single element, but this overhead is fairly minimal (figure 5.4a). The interleaved memory layout has more trouble with the variable length of rows, however. In order to transfer data between rows, they should have the same layout, which means that for striping, all rows must be striped out to the the maximum length of any row, rather than the most compact striping possible for the current row. This forces the calculation of many cells which represent impossible alignments (figure 5.4b). In practice, it is not even worth shortening the loop for the first few rows (where j is less than the vector length), and the entire square matrix is filled, resulting in 2-fold overhead. This overhead is sufficiently large that is not beneficial to deploy the interleaved layout in an alignment context. Remember though, that if we extend to the scanning case, the remainder of the matrix as j increases past the maximum value of d is rectangular, and that is amenable to striping with no additional overhead. The proportional overhead for the initial portion of the alignment shrinks as j becomes very large, and thus the interleaved memory layout is useful for scanning-type algorithms.

5.4.3 Full-precision alignment

The full-precision (32-bit) implementation of CYK alignment is a re-implementation of the reference scalar version; it does not introduce any material change in the values calculated. Both alignment context and

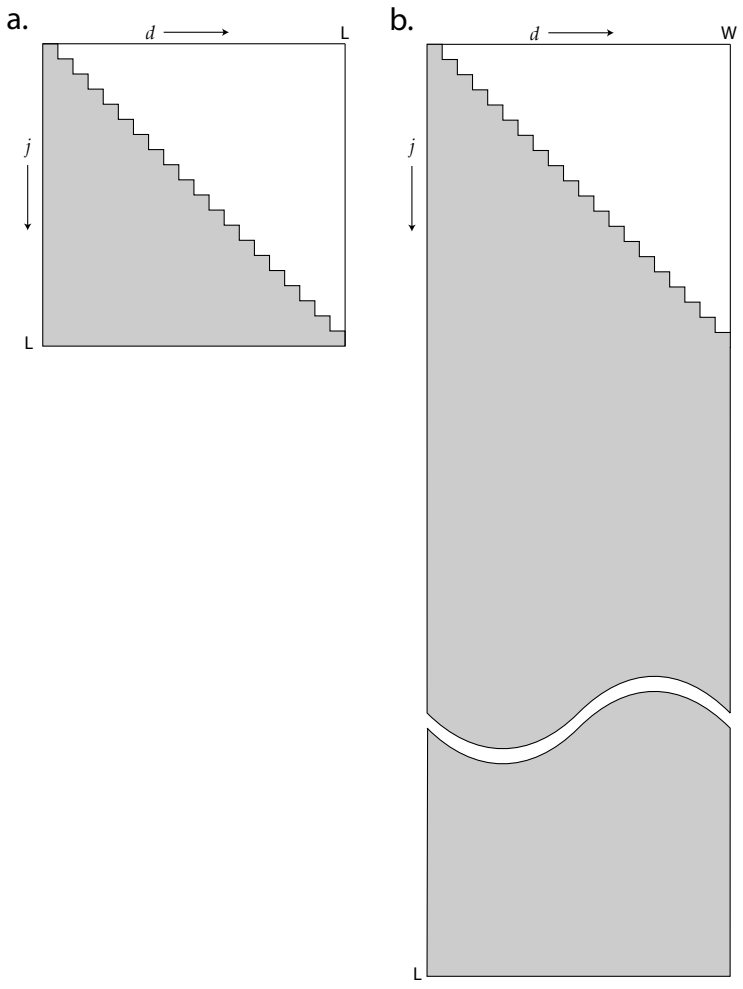


Figure 5.3: **Cell usage in alignment and scanning contexts.** Dynamic programming matrix cells in CYK alignment, invariant for a model state v. Filled (gray) areas represent possible alignments, with values that must be calculated; blank areas correspond to impossible alignments, and their values are not required. Examples are given for alignment-type CYK (a) and scanning-type (windowed) CYK (b).

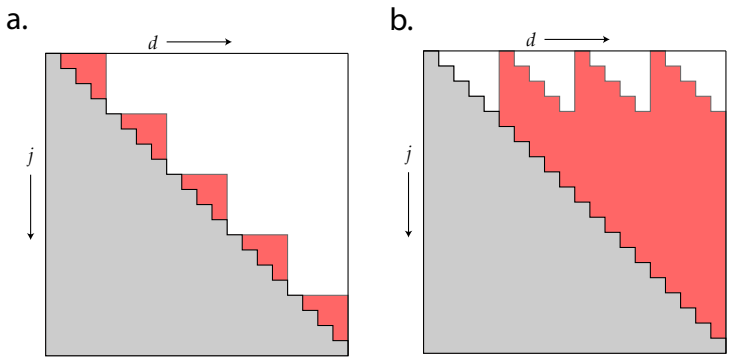


Figure 5.4: **Calculation overhead for sequential vs. interleaved layouts.** Additional matrix cells filled by parallel DP, which are overhead relative to a scalar implementation for sequential memory layout (a) and interleaved memory layout (b). Grey represents valid cells (those that must be calculated in any case), red represents overhead imposed by parallel computation.

scanning context (windowed) versions are available; as described above, the alignment context version uses sequential memory layout and the scanning context version interleaves. Running times between the two are comparable for sequences up to the window size of the model. The other major differences between them are the same as their scalar counterparts: alignment gives a single score, with optional traceback; scanning can resolve multiple hits via a semi-HMM parser, but provides only scores and coordinates. The pipeline presented here uses the scanning-type algorithm for its final stage; both versions were tested and give comparable performance, but the scanning version has more robust handling of inexact sequence coordinates and the bias composition correction term discussed in 5.5.

Using 32-bit numbers allows 4 cells to be calculated simultaneously, for a theoretical speed-up of 4-fold. This implementation does not reach that level of efficiency, empirically giving approximately 3-fold speed-up over scalar alignment. This deficit is due to overhead in setting up the parallel computation, and the dependencies within and between vectors that force calculations to wait for results that have not been finished yet. A sample of timing comparisons for scalar and vector versions of alignment type CYK is shown in figure 5.5.

5.4.4 Medium-precision alignment

The middle stage of the pipeline is fundamentally still CYK alignment, but it does introduce some changes to accommodate the numerical representation and to behave more as a filter than a full alignment engine. This stage uses 8-fold parallelism, which implies 16-bit numbers. 16-bit signed integers have an approximate range of $-32,000$ to $32,000$, and to bring these into a useful range, I scale the log-odds scores by 500 and round to integer values. This gives an effective score range of approximately -65 to 65. Although some real sequences have bitscores greater than 65, capping them at the maximum value provides enough information and they will automatically pass the filter. Rounding does introduce some error in the score, and it may accumulate over the length of the alignment, but observed errors are relatively small (generally less than 1 bit).

Alignment in the scalar reference implementation allows hits to be local with respect to the CM, but

assumes the entire sequence is to be aligned (global). Because we do not expect this stage to receive exactly correct hit boundaries to work with, I instead allow the parallel version to choose the sequence bounds of the alignment – fully local alignment – with the remainder of the sequence (unaligned) scored under a geometric length distribution. Also, because it is operating solely as a pass/fail filter, this implementation provides no mechanism for recovering a traceback of the alignment. This implementation gives an observed range of speed-up around 6-fold, slightly less than the realize the theoretically expected 8-fold, as demonstrated in figure 5.5.

5.4.5 Low-precision, ungapped alignment and search

The first stage filter also performs CYK alignment, but it is dramatically different from the later stages of the pipeline due to a modified model architecture for aligning ungapped consensus hits, and it operates in a very limited numerical range.

Numerical representation

In order to calculate 16 matrix cells simultaneously, the values in this implementation are stored as 8-bit unsigned numbers, with the very limited range of $0..255$. To make best use of this range, the corresponding floating point scores are scaled and offset before being rounded. The zero-equivalent point is set at 195 (i.e. 'positive' scores are above 195, and negative scores are below), and the unit steps are equivalent to $1/3$ bit each. This yields an equivalent range of floating point scores of $-60..20$ bits. This range is biased toward negative scores because reaching the lower bound is a 'sticky' event – it behaves as negative infinity, and any alignment which reaches that value can never recover to a higher score. Thus we want to minimize the chance of that event, and we do not actually need very large positive scores – the empirical cutoffs for 2% database passing are in the range of 5 bits.

The values in the DP matrix, each corresponding to a possible alignment, are scaled and offset as described above. The transition and emission scores from the model are also scaled in $1/3$ bit units, but are offset differently in order to represent them all as unsigned numbers. Under the rules of CM construction, all

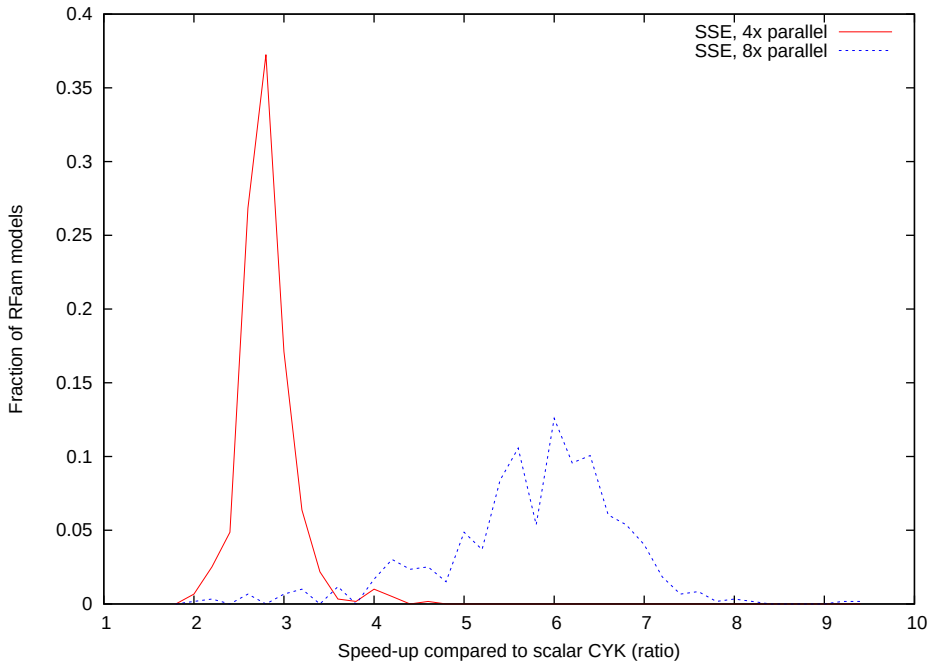


Figure 5.5: **Observed speed-up for Rfam models.** Observed speed-up for aligning sequences of various lengths to all Rfam models. Data shown are the ratio of time required for 4x (high-precision) and 8x (medium-precision) parallel versions of CYK alignment, relative to the scalar (non-parallel version). The sequence length for each model is equal to the automatically set window size of that model, corresponding to the maximum expected length of a real member of the sequence family. Times were calculated for all Rfam 9.1 models, but only searches reporting non-zero execution times (0.01 second or greater) are included here.

transition scores are negative, being logs of probabilities rather than log-odds, so transition scores are scaled and represented as positive-valued costs, which can be subtracted rather than added when calculating total alignment score. Emission (residue) scores, in contrast, are log-odds and may be positive or negative. These are offset by a bias term which is equal to the largest positive emission score anywhere in the model; the cost of a residue is then represented as the bias minus the score. Thus the highest scoring emission will be stored as a cost of zero, an emission worth 0 bits will be stored as a cost equal to the bias term, and negative scores will be stored as a positive cost greater than the bias. When accounting for a residue in the alignment, we will subtract the cost and then re-add the amount of the bias in a second step.

This arrangement of scaled and offset numbers is derived from the implementation of HMMER3's multi-segment Viterbi (MSV) filter; however there is one additional wrinkle in the conversion to a CM algorithm. The score calculation for a bifurcation (BIF) state involves adding the scores of two already calculated alignments, but note that both alignments already contain the offset zero term. Clearly, we must subtract the offset term from their sum, but the order of operations presents a quandary. If we calculate the sum first, the score will almost certainly overflow, and although we can use saturated addition to keep the score at 255, the subsequent subtraction of the offset term will give us a score much lower than we should have had. If, on the other hand, we subtract the offset from one of the alignment terms first, or even split the cost between both of them, we are likely to underflow and reach our negative infinity equivalent, from which we cannot easily recover. To address this problem, the calculation of the score at BIF states uses a specialized calculation which first adds the (offset) scores, but detects and stores the amount of any overflow, then subtracts the offset term, and finally re-adds the amount of the overflow, if it was necessary. Technical details can be found in the supplementary material in section 5.7.1.

Consensus-only, multi-hit model

As hinted at earlier, it is a change in the underlying model that makes the largest difference in the types of alignments found by the first-stage filter. The goal is to have ungapped consensus hits to the CM, but also to allow multiple of those short hits in a single alignment, with arbitrary nesting and/or sequential arrangement

of the fragments. The first step in this process is to convert the CM into a consensus-only CM, which simply means traversing the model, identifying all consensus states, which are copied to the new model, while the insertion and deletion states are ignored. Because this leaves only a single possible path through the model, all transition scores are now 0.0 (probability 1.0), and can be ignored. The emission scores for the consensus state are copied to their corresponding new state, but scaled and offset as described above.

The second step in the process is to create a few new states which will form a shell or framework around the consensus CM to allow multi-hit alignment and also account for any unaligned residues. These states are described by the grammar given in section 5.3, namely:

$$\mathrm{S} \rightarrow \mathrm{S}a|\mathrm{SM}|\epsilon \qquad\qquad \mathrm{M} \rightarrow x_{w..z}|x_{w..y}\mathrm{S}{x'}_{w..y}$$

The w, y, z indices here correspond to the states of the consensus CM, so M produces an ungapped segment of the CM, either as a wedge ($x_{w..z}$) or as a V with two sequence ranges ($x_{w..y}$ and ${x'}_{w..y}$) and a sub-alignment in between. The other state, S, produces everything else: unaligned residues ($\mathrm{S} \rightarrow \mathrm{S}a$) or another occurrence of $\mathrm{S} \rightarrow \mathrm{SM}$.

Parameterization

While full- and medium-precision alignment operate only with the fully parameterized CM, which I treat as a given for this discussion, the consensus-only ungapped model has some additional parameters which must be set. Namely, the rule

$$\mathrm{S} \rightarrow \mathrm{S}a|\mathrm{SM}|\epsilon$$

chooses between these three potential productions with probabilities t_1, t_2, and t_3, respectively, yielding 2 free parameters. The second rule

$$\mathrm{M} \rightarrow x_{w..z}|x_{w..y}\mathrm{S}{x'}_{w..y}$$

potentially has a very large number of different productions, depending on the number of valid state ranges $w..y$ and $w..z$ in the model. However, I assume a uniform distribution on model fragments, including those

extending to E states ($w..z$), which uniquely determines the probabilities for this rule, including the proportion of fragments that are terminal (wedge type, the first alternative) vs. recursive (V-type, the second alternative).

There is one additional constraint to consider when setting our free parameters, which is the expected sequence length generated by the model. The expected length is a function of the transition probabilities from non-terminal S, as well as the the length distribution of fragments generated by the model via non-terminal M. Calling the expected length of sequences generated by S $\langle n \rangle_S$, it is given by this relation:

$$\langle n \rangle_S = \frac{t_1 + t_2 \Gamma_M}{t_3 - \rho_2 t_2}$$

where Γ_M is the expected number of homologous residues emitted in a single model fragment, and ρ_2 is the proportion of fragments that do not end in E states (and thus recurse to S). For derivation, see 5.7.2. Alternatively, solving for t_2

$$t_2 = \frac{\langle n \rangle_S - t_1(\langle n \rangle_S + 1)}{\Gamma_M + \langle n \rangle_S + \rho_2 \langle n \rangle_S}$$

So, given a desired expected length, the only free parameter that remains is t_1. I set the expected length to be equal to the consensus length of the CM, which is a proxy for the expected length of the original model.

Intuitively, t_1 describes the geometric length distribution of unaligned residues - higher values imply longer stretches of unaligned residues and fewer model segments overall, and the converse is true as well. Through anecdotal testing of discrimination between real and random sequences, I set the default for t_1 to 0.25 (data not shown).

Calibration

Because the ungapped alignment model is substantially different from its underlying CM, it also has different score distribution behavior. As a result, an EVD for scores for this alignment model must be fit via simulation separately from the main CM calibration step. As shown in figure 5.6, the maximum likelihood estimates for λ values between the consensus structural alignment model and standard gapped alignment (CYK with CM model) are not well-correlated. This is in contrast to the second stage filter, which although it operates in limited precision, is close enough to the standard alignment that it can use the same calibration parameters.

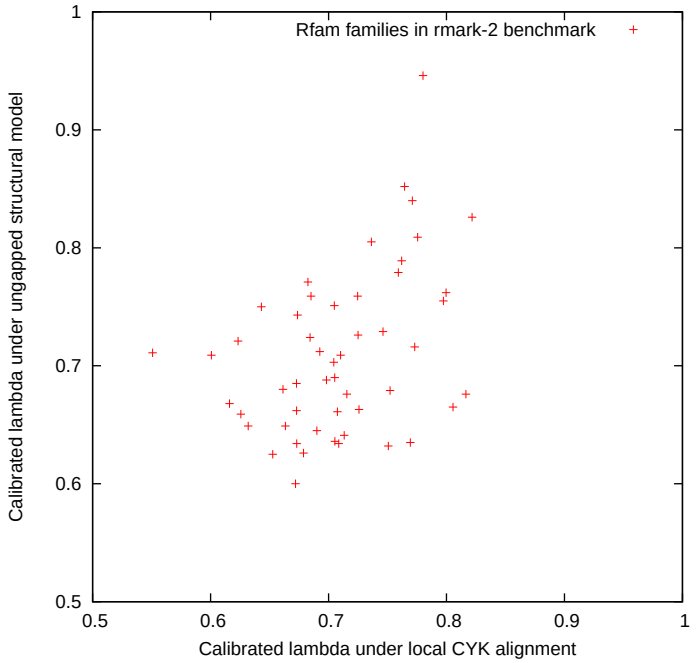


Figure 5.6: **Lambda estimates for MSCYK and standard CYK models.** Score distribution parameters are estimated separately via maximum-likelihood for standard CYK alignment and the ungapped structure alignment model, MSCYK. Values are plotted for the 51 Rfam families included in the rmark-2 benchmark; results are typical of for the remainder of Rfam families.

The calibration is sensitive to the t_1, t_2, t_3 parameters described above, and since those may be set at the per search level, calibration is also performed once per search. In principle, however, if these parameters were always fixed, then calibration could be done once per model and stored for all future searches.

Speed

Although I will discuss the overall speed characteristics of the pipeline later, it is important to consider the time requirements of this filter separately, for two reasons. First is a practical consideration: because the first filter must search the entire database, it will generally be the dominant factor in determining overall speed performance. Second, and more interestingly, it has somewhat different scaling behavior as compared to the other methods discussed here.

When compared to a reference implementation of scanning-mode CYK, the speed-ups seen for MSCYK are family-dependent, and widely varying from less than five fold for some families, up to 100 fold for others. However, they are not evenly distributed in that range; rather, the relative speed-up is a strongly-correlated function of the time required for the reference implementation (see figure 5.7).

Furthermore, if we classify the families according to number of bifurcations in the model, it is readily apparent that the best speed-ups are for those families with two or more bifurcations, while the worst correspond to families with none (i.e., unbranched hairpin structures). This result is largely expected, since the design of the MSCYK filter fixes the effective number of bifurcations at 1, regardless of the family. Calculating scores at bifurcations takes a disproportionately large amount of search time, so a large reduction in the number of bifurcations results in large reductions in time. Conversely, introducing a bifurcation to a model that previously had none cuts into the improvement that might otherwise be seen from ungapped alignment and 16-way parallelism. This might suggest that an alternative, bifurcation-free alignment approach might be best for these families. By number of families, those without bifurcations represent a large majority of models in Rfam. However, the very large families with many bifurcations dominate overall search time, and speed-up when considering searches against all families together is strongly weighted toward them.

The time complexity of the CYK algorithm in the context of CMs is generally given as $O(\mathrm{MN}^3)$ for

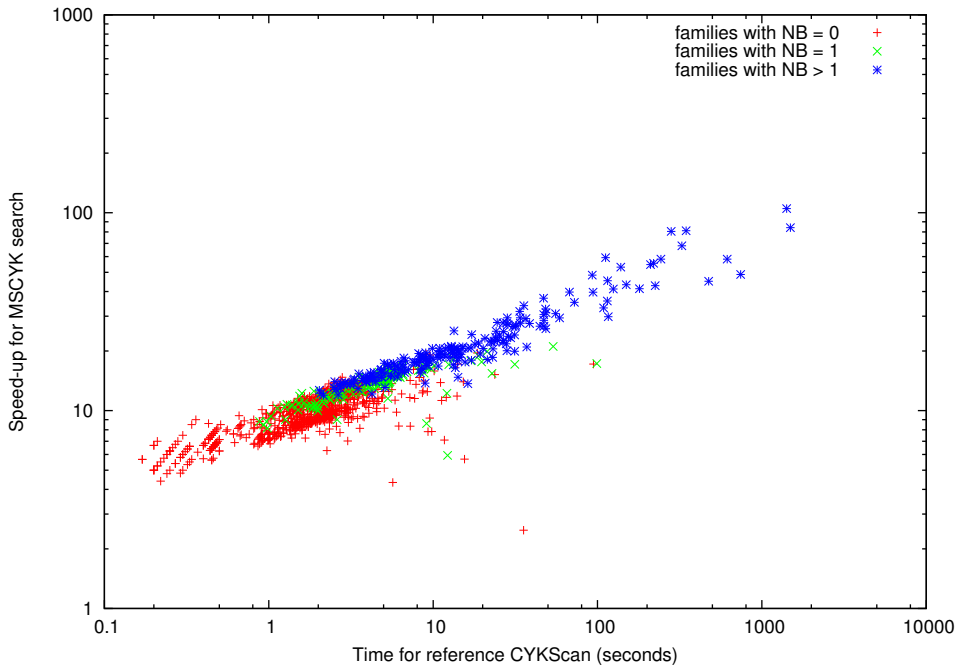


Figure 5.7: **Speed-up for MSCYK compared to time for reference CYK.** Relative speed for the MSCYK filter compared to the scalar reference algorithm CYKScan, plotted versus time required for CYKscan in seconds. Each point represents a single family from Rfam 9.1. Simulations searched 10kb of random sequence for each family.

alignments of a sequence of N residues to models of M states, or more exactly, $O(\mathrm{MN}^2 + \mathrm{BN}^3)$, with the number of bifurcation states B less than M. Scanning variants of the algorithm, however, are designed for input sequences much larger than the model size, and generally limit the length of hits to some window size W, which changes the complexity to $O(\mathrm{MNW} + \mathrm{BNW}^2)$. Knowing that the MSCYK filter fixes the number of bifurcations at 1, I conclude that the scaling behavior relative to W across all models should be the same as that for the reference algorithm on the subset of families that have $\mathrm{B} = 1$. Figure 5.8 shows the behavior for MSCYK on all families compared to the scalar version of scanning-CYK for only those families with $\mathrm{B} = 1$, and indeed, they have very similar empirical scaling relative to W.

5.4.6 Pipeline integration and settings

As described above, the first stage filter is a 'scanning' filter; it considers the entire sequence and looks for zero or more possible hits to the query model. Both the initial generation of hits and a subsequent stage of single linkage clustering allow multiple short consensus matches to be grouped into a larger hit region. This region, along with some padding on each end, defines a window that is considered in the subsequent filters. Because this search stage is calibrated against simulated random sequences, it is possible to set a P-value or E-value target for the cutoff; in practice this is implemented as a parameter describing the fraction of the sequence database that is allowed through. Because the subsequent stages are markedly slower, this parameter has the largest tunable effect on the speed of the pipeline; the recommended default based on sensitivity testing against the benchmark is 2%.

The second stage considers only the window that was previously identified, and looks for the best scoring single alignment in that window. The scores from this stage can hit a ceiling because of the limited numerical range, and this is especially common on longer families. This limits the ability of this alignment stage to accurately calculate the boundaries of the alignment; as a result the filter operates in strictly pass/fail mode, and does not narrow the window under consideration before handing it to the third stage. The default cutoff for this stage is at a P-value of 10^{-5}, determined by performance on the benchmark described in the next section. This cutoff is generally quite a bit less than the score ceiling, but it is worth noting that regardless

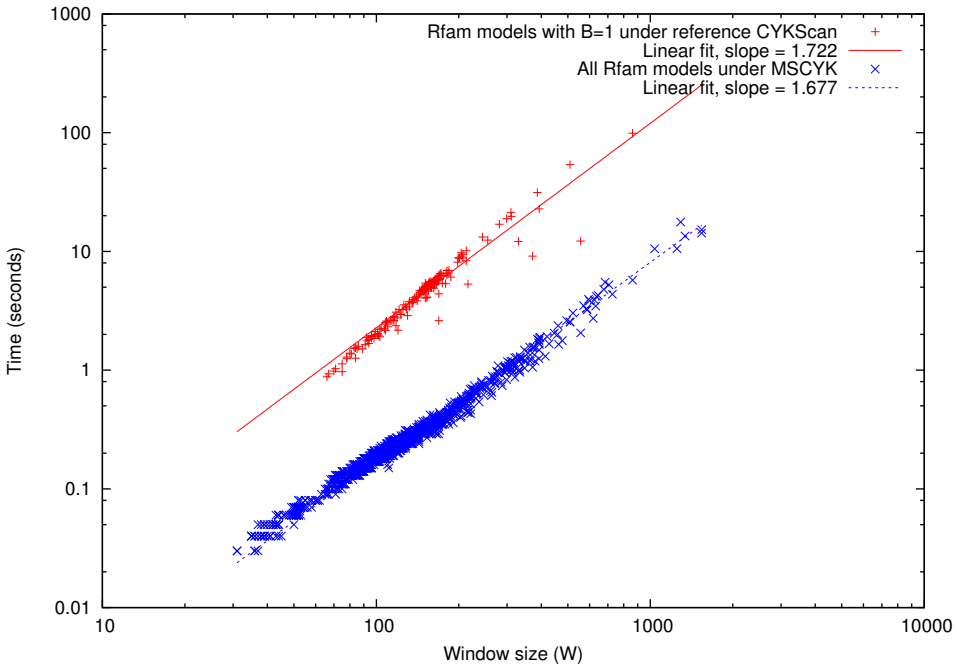


Figure 5.8: **Time complexity relative to window size.** Time in seconds for both reference CYKScan and MSCYK, versus window size W, log-log scale. The slope of the lines fit to the data indicate the exponent of the relationship, i.e., $t \propto \mathrm{W}^{1.7}$. Both datasets show very similar scaling, somewhat better than the analytical worst-case scenario of $t \propto \mathrm{W}^2$. The fits neglect the effect of the MNW term, assuming that the BNW^2 term will be dominant; N is fixed at 10000, but M varies and is roughly proportional to W.

of that cutoff, any sequences which do hit the score ceiling are automatically passed, as it is impossible to determine how much higher they could have scored.

The third and final stage re-aligns the initially identified window in full-precision mode, and determines the final bounds of the alignment. The cutoff at this stage determines how many of the results are provided to the user; the default is at an E-value of 1, although the practical cutoffs for automatic classification of homology are somewhat lower.

5.4.7 Benchmark

The benchmark used for evaluation of the software is the rmark-2 benchmark used in earlier development of the Infernal package [185, 187]. It features 450 test sequences, from 51 known ncRNA gene families, embedded in a 10 Mb 'pseudogenome'. The test sequences are separated from their corresponding alignments in such a way that they are no more than 60% identical to any sequence that remains in the set of training alignments which are used to create models of each family. The pseudogenome itself is randomly generated from an HMM which produces sequence modeled from a variety of species, such that it includes regions of biased nucleic acid composition.

Homology search methods are evaluated based on their ability to detect the embedded test sequences, given the known members of the family in the training set, while also minimizing the number of false positive hits. A reported alignment is considered a hit if the overlap between the reported sequence and the embedded sequence corresponds to at least half of the shorter of the two. Hits against a true sequence of a different family are considered to be neither true positives or false positives, because there are known to be evolutionary relationships between at least some of the ncRNA families, but these are not all well-characterized.

5.5 Evaluation

The performance of the pipeline described above on the rmark-2 benchmark is shown in figure 5.9 (blue line). Unfortunately, as this shows, its sensitivity is significantly and unacceptably lower than both unfiltered

search (red) and the currently existing filter strategy (green). The existing filter strategy starts with one round of HMM sequence-based filtering, followed by structural length-based filtering (QDB), before performing the final scoring with the Inside algorithm [187]. Although these filters do miss some sequences that would be detected without using any filters, the better than 20-fold speed-up makes the trade-off worthwhile.

The loss in sensitivity for the new filtering approach is due to one major and a few minor sources. The major loss is at the very beginning of the pipeline, at the ungapped consensus alignment stage. The alignments allowed at this stage may correspond to any portion of the model. However, there is a significant score bonus for alignments that reach the end of a hairpin with no intervening unaligned sequence. The sequences that fail to pass this stage broadly share two general features. First, they have lower than average information content (i.e., lower levels of conservation), and thus require longer alignments in order to produce a significant hit, but the requirement of ungapped, unbranched alignment limits the effective length of hits. More troubling for structural filter design, though, is the second characteristic, which is that they have insertions or deletions at all (or nearly all) of the stem-loops in the model. This is not inherently surprising; the ends of stem-loops commonly show less conservation than other portions of the model and are more amenable to changes in length. However, without at least one well-conserved stem-loop to anchor or seed the potential hit, the scores of these sequences are well down into the noise of scores from alignments of random sequences, and no reasonable cutoff threshold will retrieve them under this model of alignment.

A few sequences are also lost at later stages of the pipeline; one culprit is the application of a bias-composition correction term. A covariance model (or any other model using log-odds scores) has an explicit or implicit assumption about the composition of random sequences, usually called the background model. If the background model does not match the characteristics of the sequence being searched, alignments may be given inappropriately high scores merely for being unlikely to have been generated under the background model, rather than for being particularly likely to have been generated by the model of homology. RNA alignments have particular trouble with biased-composition sequences because AU rich or GC rich regions can also more easily be folded into structures with canonical base pairs. To counteract these problems, Infernal employs a bias-composition correction term which, in effect, replaces the standard background model

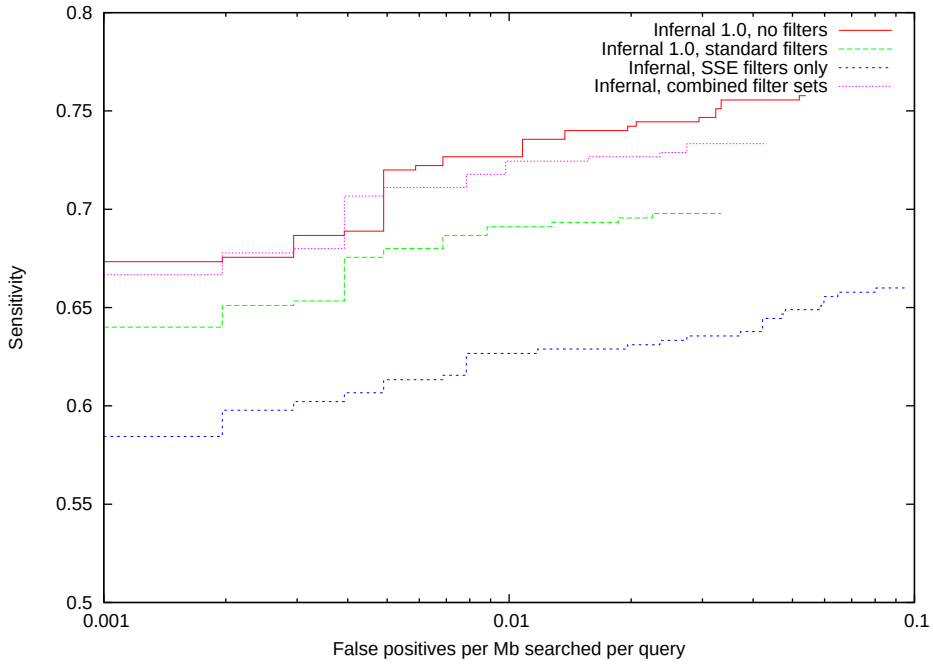


Figure 5.9: **ROC curves for the rmark-2 benchmark.** Performance plots are shown for the previously released 1.0 version of Infernal, with and without the standard filtering strategy, and the new structure-based parallel filtering strategy both alone and in combination with the existing filters. All methods use default parameters for their filter cutoff values; the curve reflects only the E-value-based sort order of the final output. All methods perform better than sequence-only alignment methods such as HMMER and BLAST (data not shown).

(which has an even distribution of bases) with a new background model which matches the observed base frequencies in the alignment under consideration. This correction term is applied *post hoc*, but it can still be included either before or after the coordinates for the alignment are determined. The final, full-precision alignment stage applies the correction to the entire matrix, before determining the bounds of the alignment. However the correction term is not particularly amenable to calculation in the lower resolution numbers used in the first and second stages. In the second, medium-precision stage in particular, the bounds of the alignment are determined before the bias-composition correction is added. In a few cases, the correction term means that the overall score does not meet the required threshold, although there was an alignment with a lower uncorrected score but a smaller correction penalty which would have passed the threshold. Although it might be possible to incorporate the correction term more directly into the filter, before determining coordinates, the correction itself is somewhat of a crude tool, and in the long term it will be better to incorporate more accurate models of background sequence in the first place.

The time required to run the structure-based filtering pipeline is on the same order as, but somewhat slower than the existing filter strategies they are being compared to. The benchmark takes approximately 23.5 CPU hours to run under Infernal 1.0 with the default filters; the first-stage structural filter alone here requires 25.4 CPU hours, with the second and third stages combining to require about half as much time again, for a total run time in the range of 37 CPU hours. Times are aggregate results with each search being run on a single 2.66 GHz Intel X5550 processor.

As as replacement for the filters that are currently used, the structure-based, parallel implementation filters I have designed are clearly a disappointment. Sensitivity is lower at comparable false positive rates, and speed is not even competitive rather than being an improvement. However, the set of sequences that are identified is not a subset of those found with the previously existing filters. Although the vast majority of sequences overlap between the two approaches, each finds some that the other does not. In particular, the approach which I have described here is able to retrieve the majority of the sequences which were considered acceptable or necessary losses in the design of the earlier approach.

Because of the apparent complementarity between the two approaches, I have designed a small driver

script which runs both pipelines, determines a union set of their results, and presents a sorted and unified output. This combined approach has sensitivity and false positive results which are competitive with running Infernal without any filters at all (figure 5.9, dashed pink line).

The running time for the combined approach is the sum of the time required for each pipeline, plus some overhead. In the proof-of-concept runs, the benchmark run took 61.2 CPU hours, still vastly better than the 648 CPU hours required when running the search without any filters. This gives only a rough upper bound on the time required if a pipeline was designed from the ground up to use both methods; since the majority of sequences are identified in common by both approaches, many alignments are unnecessarily performed two or three times, in addition to the simple overhead of reading the files multiple times, and processing and combining the two sets of hits in a slower scripting language.

5.6 Discussion

From these results, how do we judge the filters described here, particularly the first-stage consensus structural filter? At the outset, the goal was to design filters that would replace the existing sequence-based filters, rather than supplement them, and in that regard the results are a disappointment, with lower sensitivity at the same false positive rates. However, they have shown some utility in recognizing a group of sequences that would otherwise be missed. The union set of both pipeline methods has sensitivity approaching that of unfiltered search; although there are some sequences which are not found by either method, they are few in number. The performance gains in sensitivity relative to the existing filters are sufficiently large that we will generally be willing to accept the trade-off of slower speed in the combined approach. The structure-based filters take approximately twice the time of the default filters, and so the slowdown for the naive combination used here is about three-fold. It is notable, though, that the structure-based filters are within the same order of magnitude of speed as the method starting with an HMM-based filter, despite not using any steps based solely on primary sequence.

The current work does not address how *best* to integrate the two approaches, but merely demonstrates that

it is possible. The current sequence-based filters in Infernal are built on HMMER2, and it is likely that in the near future they will be migrated to the newer HMMER3 libraries, and at that time there will probably also be significant rearrangement to the filtering pipeline. It would clearly be beneficial to take the union of identified hits earlier in the processing pipeline; as mentioned earlier many sequences are found by both approaches, and in the current set-up they receive 'final' alignment not once but three times. (Once in each separate pipeline, and once more after the results lists have been merged.) One idea is to use the structural filter as a 'second-chance' filter, for sequences which are below the normal recognition threshold for a first-pass HMM-based filter, but still in a twilight zone of possibly interesting scores. Furthermore, the full-precision 4x parallel versions of CYK will likely replace instances where full (non-banded) scalar alignment is currently used, for a moderate gain in speed.

5.7 Supplementary Material

5.7.1 Handling of overflow on scores of BIF states

In the low-precision representation used by the MSCYK algorithm, alignment scores are scaled and offset. At bifurcation states, the score is the sum of the scores for two previously calculated alignments, which must be adjusted by subtracting one of the two offset terms. Depending on the order of operations, this calculation is highly likely to overflow or underflow, even when the correct value is well within the normal operating range. This section gives the modified calculation used to compensate for the occurrence of overflow.

Given two vectors of offset alignment scores 'left' and 'right', and the size of the offset 'base',

$\mathrm{sum} =$	$\mathrm{adds}(\mathrm{left}, \mathrm{right})$	Normal, saturated addition
$\mathrm{mask} =$	$\mathrm{cmpeq}(\mathrm{sum}, 255)$	Create mask of saturated positions in vector
$\mathrm{sum} =$	$\mathrm{subs}(\mathrm{sum}, \mathrm{base})$	Both left and right included the offset, so subtract once
$\mathrm{crx} =$	$\mathrm{add}(\mathrm{add}(\mathrm{left}, \mathrm{right}), 1)$	Correction term, unsaturated addition
$\mathrm{crx} =$	$\mathrm{and}(\mathrm{crx}, \mathrm{mask})$	Mask the overflow correction term
$\mathrm{sum} =$	$\mathrm{adds}(\mathrm{sum}, \mathrm{crx})$	Add final correction term

5.7.2 Derivation of expected length

The parameterization of the MSCYK model is designed such that a particular expected length for hits can be set; this section gives the derivation of the expected length of sequences generated by the model, in terms of the other parameters.

Let $\Phi^{\mathrm{X}}(n)$ be the probability of non-terminal X emitting a sequence of n residues, with $\sum_{n=0}^{\infty} \Phi^{\mathrm{X}}(n) = 1$. Then for the grammar

$$\mathrm{S} \rightarrow \mathrm{S}a|\mathrm{SM}|\epsilon \qquad\qquad \mathrm{M} \rightarrow x_{w..z}|x_{w..y}\mathrm{S}x'_{w..y}$$

with transition probabilities t_1, t_2, t_3 for the first rule and ρ_1, ρ_2 for the second,

$$\Phi^{\mathrm{S}}(n) = t_1\Phi^{\mathrm{S}}(n-1) + t_2\sum_{k=0}^{n}\Phi^{\mathrm{S}}(k)\Phi^{\mathrm{M}}(n-k) + t_3\delta_{(n=0)}$$

$$\Phi^{\mathrm{M}}(n) = \rho_1\gamma^{\mathrm{M1}}(n) + \rho_2\sum_{m=0}^{n}\gamma^{\mathrm{M2}}(m)\Phi^{\mathrm{S}}(n-m)$$

where $\gamma(n)$ is the probability of a homologous fragment of n residues. $\gamma^{\mathrm{M1}}(n)$ refers specifically to the fragments generated by the first alternative (wedge-type matches), and similarly for $\gamma^{\mathrm{M2}}(n)$ and the second alternative (V-type matches). $\gamma^{\mathrm{M2}}(n)$ is only over the length of homologous (CM-match) residues produced in the fragment, and does not include the additional length contributed by the recurrence to S. Then

$$\begin{aligned}
\langle n\rangle_{\mathrm{S}} = & \sum_{n=0}^{\infty} n\Phi^{\mathrm{S}}(n) \\
= & \; t_1\sum_{n=0}^{\infty} n\Phi^{\mathrm{S}}(n-1) + t_2\sum_{n=0}^{\infty}\sum_{k=0}^{n} n\Phi^{\mathrm{S}}(k)\Phi^{\mathrm{M}}(n-k) + t_3\sum_{n=0}^{\infty} n\delta_{(n=0)} \\
= & \; t_1\sum_{n=0}^{\infty} n\Phi^{\mathrm{S}}(n-1) + t_2\sum_{n=0}^{\infty}\sum_{k=0}^{n} n\Phi^{\mathrm{S}}(k)\left[\rho_1\gamma^{\mathrm{M1}}(n-k) + \rho_2\sum_{m=0}^{n-k}\gamma^{\mathrm{M2}}(m)\Phi^{\mathrm{S}}(n-k-m)\right] \\
& + t_3 * 0 \\
= & \; t_1\sum_{n=0}^{\infty} n\Phi^{\mathrm{S}}(n-1) + t_2\rho_1\sum_{n=0}^{\infty}\sum_{k=0}^{n} n\Phi^{\mathrm{S}}(k)\gamma^{\mathrm{M1}}(n-k) \\
& + t_2\rho_2\sum_{n=0}^{\infty}\sum_{k=0}^{n} n\Phi^{\mathrm{S}}(k)\sum_{m=0}^{n-k}\gamma^{\mathrm{M2}}(m)\Phi^{\mathrm{S}}(n-k-m)
\end{aligned}$$

With re-ordering of summation indices:

$$\begin{aligned}
\langle n\rangle_{\mathrm{S}} = \quad & t_1 \sum_{n=0}^{\infty} n\Phi^{\mathrm{S}}(n-1) + t_2\rho_1 \sum_{k=0}^{\infty}\sum_{n=k}^{\infty} n\Phi^{\mathrm{S}}(k)\gamma^{\mathrm{M1}}(n-k) \\
& + t_2\rho_2 \sum_{m=0}^{\infty}\sum_{k=0}^{\infty}\sum_{n=k+m}^{\infty} n\Phi^{\mathrm{S}}(k)\gamma^{\mathrm{M2}}(m)\Phi^{\mathrm{S}}(n-k-m) \\
= \quad & t_1 \sum_{n=0}^{\infty} n\Phi^{\mathrm{S}}(n-1) + t_2\rho_1 \sum_{k=0}^{\infty}\Phi^{\mathrm{S}}(k)\sum_{n=k}^{\infty} n\gamma^{\mathrm{M1}}(n-k) \\
& + t_2\rho_2 \sum_{m=0}^{\infty}\gamma^{\mathrm{M2}}(m)\sum_{k=0}^{\infty}\Phi^{\mathrm{S}}(k)\sum_{n=k+m}^{\infty} n\Phi^{\mathrm{S}}(n-k-m)
\end{aligned}$$

And change of variables

$$n' = n-1 \qquad\qquad n'' = n-k \qquad\qquad n''' = n-k-m$$

$$\begin{aligned}
\langle n\rangle_{\mathrm{S}} = \quad & t_1 \sum_{n'=-1}^{\infty}(n'+1)\Phi^{\mathrm{S}}(n') + t_2\rho_1 \sum_{k=0}^{\infty}\Phi^{\mathrm{S}}(k)\sum_{n''=0}^{\infty}(n''+k)\gamma^{\mathrm{M1}}(n'') \\
& + t_2\rho_2 \sum_{m=0}^{\infty}\gamma^{\mathrm{M2}}(m)\sum_{k=0}^{\infty}\Phi^{\mathrm{S}}(k)\sum_{n'''=0}^{\infty}(n'''+k+m)\Phi^{\mathrm{S}}(n''') \\
= \quad & t_1\left(\sum_{n'=0}^{\infty} n'\Phi^{\mathrm{S}}(n') + \sum_{n'=0}^{\infty}\Phi^{\mathrm{S}}(n')\right) \\
& + t_2\rho_1 \sum_{k=0}^{\infty}\Phi^{\mathrm{S}}(k)\left(\sum_{n''=0}^{\infty} n''\gamma^{\mathrm{M1}}(n'') + k\sum_{n''=0}^{\infty}\gamma^{\mathrm{M1}}(n'')\right) \\
& + t_2\rho_2 \sum_{m=0}^{\infty}\gamma^{\mathrm{M2}}(m)\sum_{k=0}^{\infty}\Phi^{\mathrm{S}}(k)\left(\sum_{n'''=0}^{\infty} n'''\Phi^{\mathrm{S}}(n''') + k\sum_{n'''=0}^{\infty}\Phi^{\mathrm{S}}(n''') + m\sum_{n'''=0}^{\infty}\Phi^{\mathrm{S}}(n''')\right)
\end{aligned}$$

The expression $\sum_{n''=0}^{\infty} n''\gamma^{\mathrm{M1}}(n'')$ is merely the expected length of type 1 fragments Γ_{M1}. Similarly, Γ_{M2} is the expected length for type 2 fragments, and the overall expected length of fragments is $\Gamma_{\mathrm{M}} = \rho_1\Gamma_{\mathrm{M1}} + \rho_2\Gamma_{\mathrm{M2}}$.

All of these values may be calculated directly from the consensus-only covariance model.

$$
\begin{aligned}
\langle n\rangle_{\mathrm{S}} = \quad & t_1\left(\langle n\rangle_{\mathrm{S}}+1\right) \\
& + t_2\rho_1 \sum_{k=0}^{\infty} \Phi^{\mathrm{S}}(k)\left(\Gamma_{\mathrm{M1}}+k\right) \\
& + t_2\rho_2 \sum_{m=0}^{\infty} \gamma^{\mathrm{M2}}(m) \sum_{k=0}^{\infty} \Phi^{\mathrm{S}}(k)\left(\langle n\rangle_{\mathrm{S}}+k+m\right) \\
= \quad & t_1\langle n\rangle_{\mathrm{S}} + t_1 \\
& + t_2\rho_1 \left(\Gamma_{\mathrm{M1}} \sum_{k=0}^{\infty} \Phi^{\mathrm{S}}(k) + \sum_{k=0}^{\infty} k\Phi^{\mathrm{S}}(k)\right) \\
& + t_2\rho_2 \sum_{m=0}^{\infty} \gamma^{\mathrm{M2}}(m) \left(\langle n\rangle_{\mathrm{S}} \sum_{k=0}^{\infty} \Phi^{\mathrm{S}}(k) + \sum_{k=0}^{\infty} k\Phi^{\mathrm{S}}(k) + m\sum_{k=0}^{\infty} \Phi^{\mathrm{S}}(k)\right) \\
= \quad & t_1\langle n\rangle_{\mathrm{S}} + t_1 + t_2\rho_1\left(\Gamma_{\mathrm{M1}}+\langle n\rangle_{\mathrm{S}}\right) + t_2\rho_2 \sum_{m=0}^{\infty} \gamma^{\mathrm{M2}}(m)\left(\langle n\rangle_{\mathrm{S}}+\langle n\rangle_{\mathrm{S}}+m\right) \\
= \quad & t_1\langle n\rangle_{\mathrm{S}} + t_1 + t_2\rho_1\Gamma_{\mathrm{M1}} + t_2\rho_1\langle n\rangle_{\mathrm{S}} + t_2\rho_2\left(2\langle n\rangle_{\mathrm{S}} + \sum_{m=0}^{\infty} m\gamma^{\mathrm{M2}}(m)\right) \\
= \quad & t_1\langle n\rangle_{\mathrm{S}} + t_1 + t_2\rho_1\Gamma_{\mathrm{M1}} + t_2\rho_1\langle n\rangle_{\mathrm{S}} + 2t_2\rho_2\langle n\rangle_{\mathrm{S}} + t_2\rho_2\Gamma_{\mathrm{M2}} \\
= \quad & t_1\langle n\rangle_{\mathrm{S}} + t_1 + t_2\left(\rho_1\Gamma_{\mathrm{M1}} + \rho_2\Gamma_{\mathrm{M2}}\right) + t_2(\rho_1+\rho_2)\langle n\rangle_{\mathrm{S}} + t_2\rho_2\langle n\rangle_{\mathrm{S}} \\
\langle n\rangle_{\mathrm{S}} = \quad & t_1\langle n\rangle_{\mathrm{S}} + t_1 + t_2\Gamma_{\mathrm{M}} + t_2\langle n\rangle_{\mathrm{S}} + t_2\rho_2\langle n\rangle_{\mathrm{S}}
\end{aligned}
$$

$$
\begin{aligned}
\langle n\rangle_{\mathrm{S}} - t_1\langle n\rangle_{\mathrm{S}} - t_2\langle n\rangle_{\mathrm{S}} - t_2\rho_2\langle n\rangle_{\mathrm{S}} &= t_1 + t_2\Gamma_{\mathrm{M}} \\
\langle n\rangle_{\mathrm{S}}(1 - t_1 - t_2 - t_2\rho_2) &= t_1 + t_2\Gamma_{\mathrm{M}} \\
\langle n\rangle_{\mathrm{S}} &= \frac{t_1 + t_2\Gamma_{\mathrm{M}}}{t_3 - t_2\rho_2}
\end{aligned}
$$

Chapter 6

Conclusions and future directions

One of the key themes of this work has been the importance of matching our computational models to the type of data we expect or the nature of the solution we hope to find. Every model encodes assumptions, both explicit and implicit, about the relationships that it describes. The creation or design of the model almost always involved trade-offs between simplicity of description and the variety of conditions which are accounted for. The considerations are not inherently problematic, but they do emphasize the caution that must be used when extending an existing model to deal with a new or slightly different case. If the new application violates some of our previous assumptions, as the alignment of fragmentary sequences from chapter 2 did, we need to readjust those assumptions to accommodate the new scenario.

The future development of covariance models includes many open avenues of research; most of the theoretical and technical limitations described in chapter 1 remain. Despite work on computation speed by both Eric Nawrocki and myself, as well as contributions from other research groups, large-scale deployments of CMs still use BLAST-based pre-filters, and eliminating the necessity of that step is a major goal of Infernal development.

In the short term, the next major revision of Infernal will likely migrate to the use of HMMER3 for its HMM-based filtering steps; this depends on currently ongoing work to extend HMMER's current focus on protein sequences to equivalently high performance on nucleic acids. This change is expected to produce

significant additional speed in Infernal, but also represents an opportunity to better integrate some of the methods I have described here. This is particularly true for the alternative filtering methods described in chapter 5. The structural filters are expected to be an important part of maintaining high sensitivity in the main search pipeline. Placing that stage behind a much faster initial filter, even one with only moderate specificity, could mitigate the remaining speed problems.

Because of its high computational requirements, we do not currently plan to add trCYK to the main processing pipeline. However, its ideas for the nature of local alignment are already being incorporated in combination with HMM-based pre-alignment to produce fragment-specific CMs. This is an optional configuration, but it has been particularly helpful in dealing with ribosomal alignments, where the sequences are frequently derived from internal PCR primer sites, thus leaving out parts of the structure. It remains to be seen how effective this approach will be on ncRNAs that have less conservation than the highly constrained ribosomal sequences.

Looking toward the longer-term future of development with covariance models, it is clear that there is a great deal of room for additional work. Suggestions for increasing the speed of CM search include formulating a separate processing pipeline for handling non-branching models. Currently, roughly 60% of RNA models in Rfam are simple hairpins – many are predicted microRNAs or snoRNAs – and although these models do not dominate overall search times because of their generally small size, the absence of bifurcations allows some approaches for speed-up that are not available to more general models. Another avenue for increasing speed would be to directly combine some of the banding-based search reduction techniques with the low-level parallelism used here. Although having a dynamic programming matrix with an arbitrary shape rather than a cube of even dimensions would present some technical difficulty for memory organization, the approaches are essentially orthogonal and could be used simultaneously rather than in sequential filters.

Further increases in speed for the MSCYK filter probably depend on simplifying the model, as it has already reached the lower limit of data representation and cannot be moved to higher parallelism on current hardware. The current model allows complex nested and sequential arrangements of ungapped segments, but it is possible that alignments of real sequences do not require this level of complexity, and might also be

recognizable for the purposes of a filter under a model which was only nested or only sequential.

The question of unstructured long insertions could be handled in a number of ways. One particularly attractive one would be to use branching models, not in the sense of the defined branching of our current models for multi-loop structures, but in the sense of probabilistic switching between alternative guide trees corresponding to alternative structures. In this way, for instance, a clade-specific insertion could be annotated with an appropriate structure, but that whole element would be considered an optional part of the model, with an overall rate of occurrence. There are other possible approaches to handling long insertions, including possibly using *de novo* folding techniques, but allowing alternative structures would do well at handling cases with multiple known structures, and might allow the consolidation of families which currently require multiple separate models for different parts of the phylogenetic tree.

Beyond the issues that have been discussed so far, the advancements that need to be made are somewhat unknown. Many of the best opportunities for improving computational methods come from identifying problems and use cases where the current approaches clearly fail. The time and memory requirements for covariance models have limited their adoption in the field, and thus limited the feedback about areas that need improvement. As the speed of the methods improve, it seems likely that increased use will highlight other areas for future work.

Bibliography

[1] B. Alpern, L. Carter, and K. S. Gatlin. Microparallelism and high-performance protein matching. In *Supercomputing '95: Proceedings of the 1995 ACM/IEEE conference on Supercomputing (CDROM)*. ACM, 1995.

[2] S. Altman and L. Kirsebom. Ribonuclease P. In R. F. Gesteland, T. R. Cech, and J. F. Atkins, editors, *The RNA World, Second Edition*, pages 351–380. Cold Spring Harbor Laboratory Press, New York, 1999.

[3] S. F. Altschul. Amino acid substitution matrices from an information theoretic perspective. *J. Mol. Biol.*, 219:555–565, 1991.

[4] S. F. Altschul, R. Bundschuh, R. Olsen, and T. Hwa. The estimation of statistical parameters for local alignment score distributions. *Nucl. Acids Res.*, 29:351–361, 2001.

[5] S. F. Altschul and W. Gish. Local alignment statistics. *Meth. Enzymol.*, 266:460–480, 1996.

[6] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *J. Mol. Biol.*, 215:403–410, 1990.

[7] S. F. Altschul, T. L. Madden, A. A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. *Nucl. Acids Res.*, 25:3389–3402, 1997.

[8] S. Altuvia, A. Zhang, L. Argaman, A. Tiwari, and G. Storz. The *Escherichia coli* OxyS regulatory RNA represses *fhlA* translation by blocking ribosome binding. *EMBO J.*, 17:6069–6075, 1998.

[9] V. Ambros. microRNAs: tiny regulators with great potential. *Cell*, 107:823–826, 2001.

[10] M. Andronescu, A. Condon, H. H. Hoos, D. H. Mathews, and K. P. Murphy. Efficient parameter estimation for RNA secondary structure prediction. *Bioinformatics*, 23:i19–i28, 2007.

[11] M. Andronescu, Z. C. Zhang, and A. Condon. Secondary structure prediction of interacting RNA molecules. *J. Mol. Biol.*, 345:987–1001, 2005.

[12] J. P. Bachellerie, J. Cavaille, and A. Hüttenhofer. The expanding snoRNA world. *Biochimie*, 84:775–790, 2002.

[13] R. Backofen and S. Will. Local sequence-structure motifs in RNA. *J. Bioinform. Comput. Biol.*, 2:681–698, 2004.

[14] V. Bafna and S. Zhang. FastR: fast database search tool for non-coding RNA. *Proc. IEEE Comput. Syst. Bioinform. Conf.*, pages 52–61, 2004.

[15] N. Ban, P. Nissen, J. Hansen, P. B. Moore, and T. A. Steitz. The complete atomic structure of the large ribosomal subunit at 2.4 Å resolution. *Science*, 289:905–920, 2000.

[16] D. P. Bartel and C. Z. Chen. Micromanagers of gene expression: the potentially widespread influence of metazoan microRNAs. *Nat. Rev. Genet.*, 5:396–400, 2004.

[17] R. T. Batey, R. P. Rambo, L. Lucast, B. Rha, and J. A. Doudna. Crystal structure of the ribonucleoprotein core of the signal recognition particle. *Science*, 287:1232–1239, 2000.

[18] L. E. Baum. An equality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. *Inequalities*, 3:1–8, 1972.

[19] P. S. Bazeley, V. Shepelev, Z. Talebizadeh, M. G. Butler, L. Fedorova, V. Filatov, and A. Fedorov. snoTARGET shows that human orphan snoRNA targets locate close to alternative splice junctions. *Gene*, 408:172–179, 2008.

[20] M. D. Been and G. S. Wickham. Self-cleaving ribozymes of hepatitis delta virus RNA. *Eur. J. Biochem.*, 247:741–753, 1997.

[21] E. H. Blackburn. Telomerase. In R. F. Gesteland, T. R. Cech, and J. F. Atkins, editors, *The RNA World, Second Edition*, pages 609–635. Cold Spring Harbor Laboratory Press, New York, 1999.

[22] S. Blanquart and N. Lartillot. A site and time-heterogeneous model of amino acid replacement. *Mol. Bio. Evol.*, 25:842–858, 2008.

[23] F. R. Blattner, G. Plunkett, C. A. Bloch, N. T. Perna, V. Burland, M. Riley, J. Collado-Vides, J. D. Glasner, C. K. Rode, G. F. Mayhew, J. Gregor, N. W. Davis, H. A. Kirkpatrick, M. A. Goeden, D. J. Rose, B. Mau, and Y. Shao. The complete genome sequence of *Escherichia coli* K-12. *Science*, 277:1453–1462, 1997.

[24] B. J. Blencowe. Transcription: surprising role for an elusive small nuclear RNA. *Curr. Biol.*, 12:R147–R149, 2002.

[25] J. U. Bowie, R. Luthy, and D. Eisenberg. A method to identify protein sequences that fold into a known three-dimensional structure. *Science*, 253:164–170, 1991.

[26] J. W. Brown. The ribonuclease P database. *Nucl. Acids Res.*, 27:314, 1999.

[27] M. P. Brown. Small subunit ribosomal RNA modeling using stochastic context-free grammars. *Proc. Int. Conf. Intell. Syst. Mol. Biol.*, 8:57–66, 2000.

[28] M. Brudno, C. B. Do, G. M. Cooper, M. F. Kim, E. D. Davydov, NISC Comparative Sequencing Program, E. Green, A. Sidow, and S. Batzoglou. LAGAN and multi-LAGAN: efficient tools for large-scale multiple alignment of genomic DNA. *Genome Res.*, 13:721–731, 2003.

[29] R. Bundschuh. Rapid significance estimation in local sequence alignment with gaps. *J Comput. Biol.*, 9:243–260, 2002.

[30] C. Burge and S. Karlin. Prediction of complete gene structures in human genomic DNA. *J. Mol. Biol.*, 268:78–94, 1997.

[31] C. B. Burge, T. Tuschl, and P. A. Sharp. Splicing of precursors to mRNAs by the spliceosomes. In R. F. Gesteland, T. R. Cech, and J. F. Atkins, editors, *The RNA World, Second Edition*, pages 525–560. Cold Spring Harbor Laboratory Press, Cold Spring Harbor, New York, 1999.

[32] G. Burns, R. Daoud, and J. Vaigl. LAM: An Open Cluster Environment for MPI. In *Proceedings of Supercomputing Symposium*, pages 379–386, 1994.

[33] S. E. Butcher, T. Dieckmann, and J. Feigon. Solution structure of a GAAA tetraloop receptor RNA. *EMBO J.*, 16:7490–7499, 1997.

[34] S. E. Butcher, T. Dieckmann, and J. Feigon. Solution structure of the conserved 16S-like ribosomal RNA UGAA tetraloop. *J. Mol. Biol.*, 268:348–358, 1997.

[35] J. J. Cannone, S. Subramanian, M. N. Schnare, J. R. Collett, L. M. D'Souza, Y. Du, B. Feng, N. Lin, L. V. Madabusi, K. M. Müller, N. Pande, Z. Shang, N. Yu, and R. R. Gutell. The comparative RNA web (CRW) site: an online database of comparative sequence and structure information for ribosomal, intron, and other RNAs. *BMC Bioinformatics*, 3:2, 2002.

[36] P. P. Chan and T. M. Lowe. GtRNAdb: a database of transfer RNA genes detected in genomic sequence. *Nucl. Acids Res.*, 37:D93–D97, 2009.

[37] M. T. Cheah, A. Wachter, N. Sudarsan, and R. R. Breaker. Control of alternative RNA splicing and gene expression by eukaryotic riboswitches. *Nature*, 447:497–500, 2007.

[38] K. Chen and L. Pachter. Bioinformatics for whole-genome shotgun sequencing of microbial communities. *PLoS Comput. Biol.*, 1:106–112, 2005.

[39] C. Cheong, G. Varani, and I. Tinoco, Jr. Solution structure of an unusually stable RNA hairpin, 5'GGAC(UUCG)GUCC. *Nature*, 346:680–682, 1990.

[40] D. K. Y. Chiu and T. Kolodziejczak. Inferring consensus structure from nucleic acid sequences. *Comput. Applic. Biosci.*, 7:347–352, 1991.

[41] C. Chothia and A. Lesk. The relation between the divergence of sequence and structure in proteins. *EMBO J.*, 5:823–826, 1986.

[42] S. Chu, R. H. Archer, J. M. Zengel, and L. Lindahl. The RNA of RNase MRP is required for normal processing of ribosomal RNA. *Proc. Natl. Acad. Sci. USA*, 91:659–663, 1994.

[43] D. A. Clayton. Nuclear gadgets in mitochondrial DNA replication and transcription. *Trends Biochem. Sci.*, 16:107–111, 1991.

[44] F. Corpet and B. Michot. RNAlign program: Alignment of RNA sequences using both primary and secondary structures. *Comput. Applic. Biosci.*, 10:389–399, 1994.

[45] F. H. C. Crick. On protein synthesis. *Symp. Soc. Exp. Biol.*, 12:138–163, 1958.

[46] F. H. C. Crick. Codon-anticodon pairing: The wobble hypothesis. *J. Mol. Biol.*, 19:548–555, 1966.

[47] K. Darty, A. Denise, and Y. Ponty. VARNA: interactive drawing and editing of the RNA secondary structure. *Bioinformatics*, 25:1974–19754, 2009.

[48] X. Darzacq, B. E. Jády, C. Verheggen, A. M. Kiss, E. Bertrand, and T. Kiss. Cajal body-specific small nuclear RNAs: a novel class of 2'-O-methylation and pseudouridylation guide RNAs. *EMBO J.*, 21:2746–2756, 2002.

[49] M. Dávila López, M. A. Rosenblad, and T. Samuelsson. Conserved and variable domains of RNase MRP RNA. *RNA Biol.*, 6:208–220, 2009.

[50] M. O. Dayhoff, R. M. Schwartz, and B. C. Orcutt. A model of evolutionary change in proteins. In M. O. Dayhoff, editor, *Atlas of Protein Sequence and Structure*, pages 345–352. National Biomedical Research Foundation, Washington DC, 1978.

[51] A. Dembo and S. Karlin. Strong limit theorems of empirical functionals for large exceedances of partial sums of I.I.D variables. *The Annals of Probability*, 19:1737–1755, 1991.

[52] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39:1–38, 1977.

[53] G. Dieci, G. Fiorino, M. Castelnuovo, M. Teichmann, and A. Pagano. The expanding RNA polymerase III transcriptome. *Trends Genet.*, 23:614–622, 2007.

[54] Y. Ding, C. Y. Chan, and C. E. Lawrence. RNA secondary structure prediction by centroids in a Boltzmann weighted ensemble. *RNA*, 11:1157–1166, 2005.

[55] Y. Ding and C. E. Lawrence. A statistical sampling algorithm for RNA secondary structure prediction. *Nucl. Acids Res.*, 31:7280–7301, 2003.

[56] C. B. Do, C. S. Foo, and S. Batzoglou. A max-margin model for efficient simultaneous alignment and folding of RNA sequences. *Bioinformatics*, 24:i68–i76, 2008.

[57] C. B. Do, D. A. Woods, and S. Batzoglou. CONTRAfold: RNA secondary structure prediction without physics-based models. *Bioinformatics*, 22:e90–e98, 2006.

[58] Y. Dou, F. Xia, and J. Jiang. Fine-grained parallel application specific computing for RNA secondary structure prediction using SCFGs on FPGA. *Proc. of the 2009 Int'l Conf. on Compilers, Architecture, and Synthesis for Embedded Systems*, pages 107–116, 2009.

[59] J. A. Doudna and R. T. Batey. Structural insights into the signal recognition particle. *Annu. Rev. Biochem.*, 73:539–557, 2004.

[60] R. D. Dowell and S. R. Eddy. Evaluation of several lightweight stochastic context-free grammars for RNA secondary structure prediction. *BMC Bioinformatics*, 5:71, 2004.

[61] R. D. Dowell and S. R. Eddy. Efficient pairwise RNA structure prediction and alignment using sequence alignment constraints. *BMC Bioinformatics*, 7:400, 2006.

[62] M. Dsouza, N. Larsen, and R. Overbeek. Searching for patterns in genomic data. *Trends Genet.*, 13:497–498, 1997.

[63] R. Durbin, S. R. Eddy, A. Krogh, and G. J. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, Cambridge UK, 1998.

[64] S. R. Eddy. RNABOB. [ftp://selab.janelia.org/pub/software/rnabob/].

[65] S. R. Eddy. Non-coding RNA genes and the modern RNA world. *Nat. Rev. Genet.*, 2:919–929, 2001.

[66] S. R. Eddy. A memory-efficient dynamic programming algorithm for optimal alignment of a sequence to an RNA secondary structure. *BMC Bioinformatics*, 3:18, 2002.

[67] S. R. Eddy. HMMER - biosequence analysis using profile hidden Markov models. [http://hmmer.janelia.org/], 2008.

[68] S. R. Eddy. A probabilistic model of local sequence alignment that simplifies statistical significance estimation. *PLoS Comput. Biol.*, 4:e1000069, 2008.

[69] S. R. Eddy. The HMMER3 user's guide. [http://hmmer.janelia.org/], 2009.

[70] S. R. Eddy and R. Durbin. RNA sequence analysis using covariance models. *Nucl. Acids Res.*, 22:2079–2088, 1994.

[71] R. C. Edgar. MUSCLE: a multiple sequence alignment method with reduced time and space complexity. *BMC Bioinformatics*, 5:113, 2004.

[72] R. C. Edgar and S. Batzoglou. Multiple sequence alignment. *Curr. Opin. Struct. Biol.*, 16:368–373, 2006.

[73] C. Ehresmann, F. Baudin, M. Mougel, P. Romby, J. P. Ebel, and B. Ehresmann. Probing the structure of RNAs in solution. *Nucl. Acids Res.*, 15:9109–9128, 1987.

[74] H. Ellingworth and S. R. Eddy. RNA structure alignment on a massively parallel computer. In B. Hertzberger and G. Serazzi, editors, *Lecture Notes in Computer Science 919: High Performance Computing and Networking*, pages 502–507, Berlin, 1995. Springer.

[75] J. C. Ellis and J. W. Brown. The RNase P family. *RNA Biol.*, 6:362–369, 2009.

[76] D. Evans, S. M. Marquez, and N. R. Pace. RNase P: interface of the RNA and protein worlds. *Trends Biochem. Sci.*, 31:333–341, 2006.

[77] M. Farrar. Striped Smith-Waterman speeds database searches six times over other SIMD implementations. *Bioinformatics*, 23:156–161, 2007.

[78] R. D. Finn, J. Tate, J. Mistry, P. C. Coggill, S. J. Sammut, H.-R. Hotz, G. Ceric, K. Forslund, S. R. Eddy, E. L. L. Sonnhammer, and A. Bateman. The Pfam protein families database. *Nucl. Acids Res.*, 36:D281–D288, 2008.

[79] R. Flores, S. Delgado, M. E. Gas, A. Carbonell, D. Molina, S. Gago, and M. De la Peña. Viroids: the minimal non-coding RNAs with autonomous replication. *FEBS Lett.*, 567:42–48, 2004.

[80] M. J. Flynn. Some computer organizations and their effectiveness. *IEEE Transactions on Computers*, c-21:948–960, 2002.

[81] S. M. Freier, R. Kierzek, J. A. Jaeger, N. Sugimoto, M. H. Caruthers, T. Neilson, and D. H. Turner. Improved free-energy parameters for predictions of RNA duplex stability. *Proc. Natl. Acad. Sci. USA*, 83:9373–9377, 1986.

[82] E. K. Freyhult, J. P. Bollback, and P. P. Gardner. Exploring genomic dark matter: A critical assessment of the performance of homology search methods on noncoding RNA. *Genome Res.*, 17:117–125, 2007.

[83] E. Gabriel, G. E. Fagg, G. Bosilca, T. Angskun, J. J. Dongarra, J. M. Squyres, V. Sahay, P. Kambadur, B. Barrett, A. Lumsdaine, R. H. Castain, D. J. Daniel, R. L. Graham, and T. S. Woodall. Open MPI: goals, concept, and design of a next generation MPI implementation. In *Proceedings, 11th European PVM/MPI Users' Group Meeting*, pages 97–104, Budapest, Hungary, September 2004.

[84] P. Ganot, M. L. Bortolin, and T. Kiss. Site-specific pseudouridine formation in preribosomal RNA is guided by small nucleolar RNAs. *Cell*, 89:799–809, 1997.

[85] P. P. Gardner, J. Daub, J. G. Tate, E. P. Nawrocki, D. L. Kolbe, S. Lindgreen, A. C. Wilkinson, R. D. Finn, S. Griffiths-Jones, S. R. Eddy, and A. Bateman. Rfam: Updates to the RNA families database. *Nucl. Acids Res.*, 37:D136–D140, 2009.

[86] D. Gautheret, S. H. Damberger, and R. R. Gutell. Identification of base-triples in RNA using comparative sequence analysis. *J. Mol. Biol.*, 248:27–43, 1995.

[87] D. Gautheret, F. Major, and R. Cedergren. Pattern searching/alignment with RNA primary and secondary structures: An effective descriptor for tRNA. *Comput. Applic. Biosci.*, 6:325–331, 1990.

[88] J. F. Gibrat, T. Madej, and S. H. Bryant. Surprising similarities in structure comparison. *Curr. Opin. Struct. Biol.*, 6:377–385, 1996.

[89] S. J. Giovannoni, T. B. Britschgi, C. L. Moyer, and K. G. Field. Genetic diversity in Sargasso Sea bacterioplankton. *Nature*, 345:60–63, 1990.

[90] P. M. Gordon, E. J. Sontheimer, and J. A. Piccirilli. Metal ion catalysis during the exon-ligation step of nuclear pre-mRNA splicing: extending the parallels between the spliceosome and group II introns. *RNA*, 6:199–205, 2000.

[91] J. Gorodkin, L. J. Heyer, and G. D. Stormo. Finding the most significant common sequence and structure motifs in a set of RNA sequences. *Nucl. Acids Res.*, 25:3724–3732, 1997.

[92] O. Gotoh. An improved algorithm for matching biological sequences. *J. Mol. Biol.*, 162:705–708, 1982.

[93] O. Gotoh. Multiple sequence alignment: algorithms and applications. *Adv. Biophys.*, 36:159–206, 1999.

[94] L. Grate. Automatic RNA secondary structure determination with stochastic context-free grammars. *Proc. Int. Conf. Intell. Syst. Mol. Biol.*, 3:136–144, 1995.

[95] L. Grate, M. Diekhans, D. Dahle, and R. Hughey. Sequence analysis with the Kestrel SIMD parallel processor. *Pac. Symp. Biocomput.*, pages 263–74, 2001.

[96] L. Grate, M. Herbster, R. Hughey, D. Haussler, I.S. Mian, and H. Noller. RNA modeling using Gibbs sampling and stochastic context free grammars. *Proc. Int. Conf. Intell. Syst. Mol. Biol.*, 2:138–146, 1994.

[97] M. Gribskov, A. D. McLachlan, and D. Eisenberg. Profile analysis: Detection of distantly related proteins. *Proc. Natl. Acad. Sci. USA*, 84:4355–4358, 1987.

[98] M. Gribskov and S. Veretnik. Identification of sequence pattern with profile analysis. *Meth. Enzymol.*, 26:198–212, 1996.

[99] S. Griffiths-Jones, A. Bateman, M. Marshall, A. Khanna, and S. R. Eddy. Rfam: an RNA family database. *Nucl. Acids Res.*, 31:439–441, 2003.

[100] S. Griffiths-Jones, S. Moxon, M. Marshall, A. Khanna, S. R. Eddy, and A. Bateman. Rfam: Annotating non-coding RNAs in complete genomes. *Nucl. Acids Res.*, 33:D121–D141, 2005.

[101] W. Gropp, E. Lusk, N. Doss, and A. Skjellum. High-performance, portable implementation of the MPI Message Passing Interface Standard. *Parallel Computing*, 22:789–828, 1996.

[102] C. Guerrier-Takada, K. Gardiner, T. Marsh, N. Pace, and S. Altman. The RNA moiety of ribonuclease P is the catalytic subunit of the enzyme. *Cell*, 35:849–857, 1983.

[103] Dan Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press, Cambridge, UK, 1997.

[104] R. R. Gutell, A. Power, G. Z. Hertz, E. J. Putz, and G. D. Stormo. Identifying constraints on the higher-order structure of RNA: Continued development and application of comparative sequence analysis methods. *Nucl. Acids Res.*, 20:5785–5795, 1992.

[105] B. Harris, A. C. Jacob, J. M. Lancaster, J. Buhler, and R. D. Chamberlain. A banded Smith-Waterman FPGA accelerator for Mercury BLASTP. *Proc. of 17th Int'l Conf. on Field Programmable Logic and Applications*, 2007.

[106] H. Hasegawa and L. Holm. Advances and pitfalls of protein structural alignment. *Curr. Opin. Struct. Biol.*, 19:341–348, 2009.

[107] J. H. Havgaard, R. B. Lyngso, G. D. Stormo, and J. Gorodkin. Pairwise local structural alignment of RNA sequences with sequence similarity less than 40%. *Bioinformatics.*, 21:1815–1824, 2005.

[108] C. S. Hayes and K. C. Keiler. Beyond ribosome rescue: tmRNA and co-translational processes. *FEBS Lett.*, 584:413–419, 2010.

[109] C. U. Hellen and P. Sarnow. Internal ribosome entry sites in eukaryotic mRNA molecules. *Genes Dev.*, 15:1593–1612, 2001.

[110] R. Hempel. The MPI Standard for message passing. In W. Gentzsch and U. Harms, editors, *High–Performance Computing and Networking, International Conference and Exhibition, Proceedings, Volume II: Networking and Tools*, volume 797, pages 247–252, 1994.

[111] J. Henderson, S. Salzberg, and K. Fasman. Finding genes in human DNA with a hidden Markov model. *J. Comput. Biol.*, 4:127–141, 1997.

[112] S. Henikoff and J. G. Henikoff. Amino acid substitution matrices from protein blocks. *Proc. Natl. Acad. Sci. USA*, 89:10915–10919, 1992.

[113] M. W. Hentze and L. C. Kühn. Molecular control of vertebrate iron metabolism: mRNA-based regulatory circuits operated by iron, nitric oxide, and oxidative stress. *Proc. Natl. Acad. Sci. USA*, 93:8175–8182, 1996.

[114] H. A. Heus and A. Pardi. Structural features that give rise to the unusual stability of RNA hairpins containing GNRA loops. *Science*, 253:191–194, 1991.

[115] M. B. Hoagland, M. L. Stephenson, J. F. Scott, L. I. Hecht, and P. C. Zamecnik. A soluble ribonucleic acid intermediate in protein synthesis. *J. Biol. Chem.*, 231:241–257, 1958.

[116] A. Hochsmann, T. Toller, R. Giegerich, and S. Kurtz. Local similarity in RNA secondary structures. *Proc. of Comput. Sys. Bioinf. (CSB 2003)*, 2:159–169, 2003.

[117] I. L. Hofacker, S. H. Bernhart, and P. F. Stadler. Alignment of RNA base pairing probability matrices. *Bioinformatics.*, 20:2222–2227, 2004.

[118] I. L. Hofacker, M. Fekete, and P. F. Stadler. Secondary structure prediction for aligned RNA sequences. *J. Mol. Biol.*, 319:1059–1066, 2002.

[119] S. R. Holbrook. Crystallographic analysis of RNA structure. In R. W. Simon and M. Grunberg-Manago, editors, *RNA Structure and Function*, pages 147–174. Cold Spring Harbor Laboratory Press, 1998.

[120] R. W. Holley, J. Apgar, G. A. Everett, J. T. Madison, M. Marquisee, S. H. Merrill, J. R. Penswick, and A. Zamir. Structure of a ribonucleic acid. *Science*, 14:1462–1465, 1965.

[121] I. Holmes. Accelerated probabilistic inference of RNA structure evolution. *BMC Bioinformatics*, 6:73, 2005.

[122] I. Holmes and R. Durbin. Dynamic programming alignment accuracy. *J. Comput. Biol.*, 5:493–504, 1998.

[123] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Reading, Massachusetts, 1979.

[124] Z. Huang, Y. Wu, J. Robertson, L. Feng, R. Malmberg, and L. Cai. Fast and accurate search for non-coding RNA pseudoknot structures in genomes. *Bioinformatics*, 24:2281–2287, 2008.

[125] A. Hüttenhofer, P. Schattner, and N. Polacek. Non-coding RNAs: hope or hype? *Trends Genet.*, 21:289–297, 2005.

[126] B. E. Jády and T. Kiss. A small nucleolar guide RNA functions both in 2'-O-ribose methylation and pseudouridylation of the U5 spliceosomal RNA. *EMBO J.*, 20:541–551, 2001.

[127] Y. Ji, X. Xu, and G. D. Stormo. A graph theoretical approach to predict common RNA secondary structure motifs including pseudoknots in unaligned sequences. *Bioinformatics.*, 20:1591–1602, 2004.

[128] T. Jiang, G. Lin, and K. Zhang. A general edit distance between RNA structures. *J. Comput. Biol.*, 9:371–388, 2002.

[129] S. Johnson. *Remote Protein Homology Detection Using Hidden Markov Models*. PhD thesis, Washington University School of Medicine, 2006.

[130] D. T. Jones, W. R. Taylor, and J. M. Thornton. A new approach to protein fold recognition. *Nature*, 358:86–89, 1992.

[131] S. Jung, E. Swart, P. Minx, V. Magrini, E. Mardis, L. Landweber, and S. R. Eddy. Exploiting *Oxytricha trifallax* nanochromosomes to screen for noncoding RNA genes. Manuscript in preparation.

[132] S. Karlin and S. F. Altschul. Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. *Proc. Natl. Acad. Sci. USA*, 87:2264–2268, 1990.

[133] T. Kasami. An efficient recognition and syntax algorithm for context-free algorithms. Technical Report AFCRL-65-758, Air Force Cambridge Research Lab, Bedford, Mass., 1965.

[134] K. C. Keiler, P. R. H. Waller, and R. T. Sauer. Role of a peptide tagging system in degradation of proteins synthesized from damaged messenger RNA. *Science*, 271:990–993, 1996.

[135] S. H. Kim. Crystal structure of yeast tRNA-Phe and general structural features of other tRNAs. In P. R. Schimmel, D. Soll, and J. N. Abelson, editors, *Transfer RNA: Structure, Properties, and Recognition*. Cold Spring Harbor Laboratory, New York, 1979.

[136] S. H. Kim, J. L. Sussman, G. J. Quigley, A. McPherson, A. H. Wang, N. C. Seeman, and A. Rich. The general structure of transfer RNA molecules. *Proc. Natl. Acad. Sci. USA*, 71:4970–4, 1974.

[137] H. Kiryu, Y. Tabei, T. Kin, and K. Asai. Murlet: a practical multiple alignment tool for structural RNA sequences. *Bioinformatics*, 23:1588–1598, 2007.

[138] S. Kishore, A. Khanna, Z. Zhang, J. Hui, P. J. Balwierz, M. Stefan, C. Beach, R. D. Nicholls, M. Zavolan, and S. Stamm. The snoRNA MBII-52 (SNORD 115) is processed into smaller RNAs and regulates alternative splicing. *Hum. Mol. Genet.*, 19:1153–1164, 2010.

[139] S. Kishore and S. Stamm. The snoRNA HBII-52 regulates alternative splicing of the serotonin receptor 2c. *Science*, 311:230–232, 2006.

[140] R. J. Klein and S. R. Eddy. RSEARCH: finding homologs of single structured RNA sequences. *BMC Bioinformatics*, 4:44, 2003.

[141] R. J. Klein, Z. Misulovin, and S. R. Eddy. Noncoding RNA genes identified in AT-rich hyperthermophiles. *Proc. Natl. Acad. Sci. USA*, 99:7542–7547, 2002.

[142] B. Knudsen and J. Hein. RNA secondary structure prediction using stochastic context-free grammars and evolutionary history. *Bioinformatics*, 15:446–454, 1999.

[143] B. Knudsen and J. Hein. Pfold: RNA secondary structure prediction using stochastic context-free grammars. *Nucl. Acids Res.*, 31:3423–3428, 2003.

[144] D. L. Kolbe and S. R. Eddy. Local RNA structure alignment with incomplete sequence. *Bioinformatics*, 25:1236–1243, 2009.

[145] R. Kolter and C. Yanofsky. Attenuation in amino acid biosynthetic operons. *Annu. Rev. Genet.*, 16:113–134, 1982.

[146] I. Korf. Gene finding in novel genomes. *BMC Bioinformatics*, 5:59–67, 2004.

[147] S.P.T. Krishnan, S. Liang, and B. Veeravalli. Towards high performance computing for molecular structure prediction using IBM Cell broadband engine - an implementation perspective. *BMC Bioinformatics*, 11:S36, 2010.

[148] E. Krissinel and K. Henrick. Secondary-structure matching (SSM), a new tool for fast protein structure alignment in three dimensions. *Acta Crystallogr. D Biol. Crystallogr.*, 60:2256–2268, 2004.

[149] A. Krogh, M. Brown, I. S. Mian, K. Sjölander, and D. Haussler. Hidden Markov models in computational biology: Applications to protein modeling. *J. Mol. Biol.*, 235:1501–1531, 1994.

[150] A. Krogh, I. S. Mian, and D. Haussler. A hidden Markov model that finds genes in *E. coli* DNA. *Nucl. Acids Res.*, 22:4768–4778, 1994.

[151] A. Kryshtafovych, K. Fidelis, and J. Moult. CASP8 results in context of previous experiments. *Proteins*, 77:217–228, 2009.

[152] E. C. Lai, P. Tomancak, R. W. Williams, and G. M. Rubin. Computational identification of *Drosophila* microRNA genes. *Genome Biol.*, 4:R42, 2003.

[153] M. M. Lai. The molecular biology of hepatitis delta virus. *Annu. Rev. Biochem.*, 64:259–286, 1995.

[154] A. M. Lambowitz and M. G. Caprara. Group I and group II ribozymes as RNPs: clues to the past and guides to the future. In R. F. Gesteland, T. R. Cech, and J. F. Atkins, editors, *The RNA World, Second Edition*, pages 451–485. Cold Spring Harbor Laboratory Press, Cold Spring Harbor, New York, 1999.

[155] K. Lari and S. J. Young. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language*, 4:35–56, 1990.

[156] R. C. Lee, R. L. Feinbaum, and V. Ambros. The *C. elegans* heterochronic gene *lin-4* encodes small RNAs with antisense complementarity to *lin-14*. *Cell*, 75:843–854, 1993.

[157] Z. M. Lee, C. Bussema, and T. M. Schmidt. rrnDB: documenting the number of rRNA and tRNA genes in Bacteria and Archaea. *Nucl. Acids Res.*, 37:D489–D493, 2009.

[158] N. B. Leontis, J. Stombaugh, and E. Westhof. The non-Watson-Crick base pairs and their associated isostericity matrices. *Nucl. Acids Res.*, 30:3497–3531, 2002.

[159] I. Letunic, T. Doerks, and P. Bork. SMART 6: recent updates and new developments. *Nucl. Acids Res.*, 37:D229–D232, 2009.

[160] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10:707–710, 1966. Original in Russian.

[161] M. Levitt. Detailed molecular model for transfer ribonucleic acid. *Nature*, 224:759–763, 1969.

[162] I. T. Li, W. Shum, and K. Truong. 160-fold acceleration of the Smith-Waterman algorithm using a field programmable gate array (FPGA). *BMC Bioinformatics*, 8:185, 2007.

[163] L. P. Lim, N. C. Lau, E. G. Weinstein, A. Abdelhakim, S. Yekta, M. W. Rhoades, C. B. Burge, and D. P. Bartel. The microRNAs of *Caenorhabditis elegans*. *Genes Dev.*, 17:991–1008, 2003.

[164] D. J. Lipman, S. F. Altschul, and J. D. Kececioglu. A tool for multiple sequence alignment. *Proc. Natl. Acad. Sci. USA*, 86:4412–4415, 1989.

[165] T. Liu and B. Schmidt. Parallel RNA secondary structure prediction using stochastic context-free grammars. *Concurrency and Computation: Practice and Experience*, 17:1669–1685, 2005.

[166] Z. Liu, M. Reches, I. Groisman, and H. Engelberg-Kulka. The nature of the minimal 'selenocysteine insertion sequence' (SECIS) in *Escherichia coli*. *Nucl. Acids Res.*, 26:896–902, 1998.

[167] T. M. Lowe and S. R. Eddy. tRNAscan-SE: A program for improved detection of transfer RNA genes in genomic sequence. *Nucl. Acids Res.*, 25:955–964, 1997.

[168] T. M. Lowe and S. R. Eddy. A computational screen for methylation guide snoRNAs in yeast. *Science*, 283:1168–1171, 1999.

[169] Z. Lygerou, C. Allmang, D. Tollerbey, and B. Séraphin. Accurate processing of a eukaryotic precursor ribosomal RNA by ribonuclease MRP in vitro. *Science*, 272:268–270, 1996.

[170] R. B. Lyngsø and C. N. Pedersen. RNA pseudoknot prediction in energy-based models. *J. Comput. Biol.*, 7:409–427, 2000.

[171] T. J. Macke, D. J. Ecker, R. R. Gutell, D. Gautheret, D. A. Case, and R. Sampath. RNAMotif, an RNA secondary structure definition and search algorithm. *Nucl. Acids Res.*, 29:4724–4735, 2001.

[172] A. Marchler-Bauer, J. B. Anderson, M. K. Derbyshire, C. DeWeese-Scott, N. R. Gonzales, M. Gwadz, L. Hao, S. He, D. I. Hurwitz, J. D. Jackson, Z. Ke, D. Krylov, C. J. Lanczycki, C. A.Liebert, C. Liu, F. Lu, S. Lu, G. H. Marchler, M. Mullokandov, J. S. Song, N. Thanki, R. A. Yamashita, J. J. Yin, D. Zhang, and S.H. Bryant. CDD: a conserved domain database for interactive domain family analysis. *Nucl. Acids Res.*, 35:D237–D240, 2007.

[173] M. Martick and W. G. Scott. Tertiary contacts distant from the active site prime a ribozyme for catalysis. *Cell*, 126:309–320, 2006.

[174] J. M. Mason and H. Biessmann. The unusual telomeres of Drosophila. *Trends Genet.*, 11:58–62, 1995.

[175] D. H. Mathews, J. Sabina, M. Zuker, and D. H. Turner. Expanded sequence dependence of thermodynamic parameters improves prediction of RNA secondary structure. *J. Mol. Biol.*, 288:911–940, 1999.

[176] D. H. Mathews and D. H. Turner. Dynalign: an algorithm for finding the secondary structure common to two RNA sequences. *J. Mol. Biol.*, 317:191–203, 2002.

[177] V. Matys, O. V. Kel-Margoulis, E. Fricke, I. Liebich, S. Land, A. Barre-Dirrie, I. Reuter, D.Chekmenev, M. Krull, K. Hornischer, N. Voss, P. Stegmaier, B. Lewicki-Potapov, H. Saxel, A. E. Kel, and E. Wingender. TRANSFAC and its module TRANSCompel: transcriptional gene regulation in eukaryotes. *Nucl. Acids Res.*, 34:D108–D110, 2006.

[178] J. S. McCaskill. The equilibrium partition function and base pair binding probabilities for RNA secondary structure. *Biopolymers*, 29:1105–19, 1990.

[179] F. Michel, A. D. Ellington, S. Couture, and J. W. Szostak. Phylogenetic and genetic evidence for base-triples in the catalytic domain of group I introns. *Nature*, 347:578–580, 1990.

[180] S. D. Moore and R. T. Sauer. The tmRNA system for translational surveillance and ribosome rescue. *Annu. Rev. Biochem.*, 76:101–124, 2007.

[181] J. Moult. A decade of CASP: progress, bottlenecks and prognosis in protein structure prediction. *Curr. Opin. Struct. Biol.*, 15:285–289, 2005.

[182] J. Moult, K. Fidelis, A. Kryshtafovych, B. Rost, and A. Tramontano. Critical assessment of methods of protein structure prediction round VIII. *Proteins*, 77:1–4, 2009.

[183] A. Murzin, S. E. Brenner, T. Hubbard, and C. Chothia. SCOP: A structural classification of proteins database for the investigation of sequences and structures. *J. Mol. Biol.*, 247:536–540, 1995.

[184] E. P. Nawrocki. *Structural RNA Homology Search and Alignment Using Covariance Models*. PhD thesis, Washington University School of Medicine, 2009.

[185] E. P. Nawrocki and S. R. Eddy. Query-dependent banding (QDB) for faster RNA similarity searches. *PLoS Comput. Biol.*, 3:e56, 2007.

[186] E. P. Nawrocki, D. L. Kolbe, and S. R. Eddy. Infernal - inference of RNA secondary structure alignments. [http://infernal.janelia.org/], 2009.

[187] E. P. Nawrocki, D. L. Kolbe, and S. R. Eddy. Infernal 1.0: Inference of RNA alignments. *Bioinformatics*, 25:1335–1337, 2009.

[188] E. P. Nawrocki, D. L. Kolbe, and S. R. Eddy. The Infernal user's guide. [http://infernal.janelia.org/], 2009.

[189] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.*, 48:443–453, 1970.

[190] C. Notredame. Recent evolutions of multiple sequence alignment algorithms. *PLoS Comput. Biol.*, 3:e123, 2007.

[191] C. Notredame, D. G. Higgins, and J. Heringa. T-Coffee: A novel method for fast and accurate multiple sequence alignment. *J. Mol. Biol.*, 302:205–217, 2000.

[192] C. Notredame, E. A. O'Brien, and D. G. Higgins. RAGA: RNA sequence alignment by genetic algorithm. *Nucl. Acids Res.*, 25:4570–4580, 1997.

[193] R. Nussinov, G. Pieczenik, J. R. Griggs, and D. J. Kleitman. Algorithms for loop matchings. *SIAM J. Appl. Math.*, 35:68–82, 1978.

[194] Y. Okada, K. Sato, and Y. Sakakibara. Improvement of structure conservation index with centroid estimators. *Pac. Symp. Biocomput.*, pages 88–97, 2010.

[195] J. Park, K. Karplus, C. Barrett, R. Hughey, D. Haussler, T. Hubbard, and C. Chothia. Sequence comparisons using multiple sequences detect three times as many remote homologues as pairwise methods. *J. Mol. Biol.*, 284:1201–1210, 1998.

[196] W. R. Pearson and D. J. Lipman. Improved tools for biological sequence comparison. *Proc. Natl. Acad. Sci. USA*, 85:2444–2448, 1988.

[197] O. Perriquet, H. Touzet, and M. Dauchet. Finding the common structure shared by two homologous RNAs. *Bioinformatics*, 19:108–116, 2003.

[198] P. Piccinelli, M. A. Rosenblad, and T. Samuelsson. Identification and analysis of ribonuclease P and MRP RNA in a broad range of eukaryotes. *Nucl. Acids Res.*, 33:4485–4495, 2005.

[199] E. V. Puglisi and J. D. Puglisi. Nuclear magnetic resonance spectroscopy of RNA. In R. W. Simon and M. Grunberg-Manago, editors, *RNA Structure and Function*, pages 117–146. Cold Spring Harbor Laboratory Press, 1998.

[200] B. Qian and R. A. Goldstein. Detecting distant homologs using phylogenetic tree-based HMMs. *Proteins*, 52:446–453, 2003.

[201] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE*, 77:257–286, 1989.

[202] J. Reeder and R. Giegerich. Consensus shapes: an alternative to the Sankoff algorithm for RNA consensus structure prediction. *Bioinformatics*, 21:3516–3523, 2005.

[203] R. Reiner, Y. Ben-Asouli, I. Krilovetzky, and N. Jarrous. A role for the catalytic ribonucleoprotein RNase P in RNA polymerase III transcription. *Genes Dev.*, 20:1621–1635, 2006.

[204] E. Rivas and S. R. Eddy. A dynamic programming algorithm for RNA structure prediction including pseudoknots. *J. Mol. Biol.*, 285:2053–2068, 1999.

[205] E. Rivas and S. R. Eddy. Secondary structure alone is generally not statistically significant for the detection of noncoding RNAs. *Bioinformatics*, 6:583–605, 2000.

[206] E. Rivas and S. R. Eddy. Noncoding RNA gene detection using comparative sequence analysis. *BMC Bioinformatics*, 2:8, 2001.

[207] C. E. Robertson, J. K. Harris, J. R. Spear, and N. R. Pace. Phylogenetic diversity and ecology of environmental Archaea. *Curr. Opin. Microbiol.*, 8:638–642, 2005.

[208] T. Rognes and E. Seeberg. Six-fold speed-up of Smith-Waterman sequence database searches using parallel processing on common microprocessors. *Bioinformatics*, 16:699–706, 2000.

[209] A. Roth and R. R. Breaker. The structural and functional diversity of metabolite-binding riboswitches. *Annu. Rev. Biochem.*, 78:305–334, 2009.

[210] D. B. Rubin. Inference and missing data. *Biometrika*, 63:581–592, 1976.

[211] P. B. Rupert and A. R. Ferré-D'Amaré. Crystal structure of a hairpin ribozyme-inhibitor complex with implications for catalysis. *Nature*, 410:780–786, 2001.

[212] D. B. Rusch, A. L. Halpern, G. Sutton, K. B. Heidelberg, S. Williamson, S. Yooseph, D. Wu, J. A. Eisen, J. M. Hoffman, K. Remington, K. Beeson, B. Tran, H. Smith, H. Baden-Tillson, C. Stewart, J. Thorpe, J. Freeman, C. Andrews-Pfannkoch, J. E. Venter, K. Li, S. Kravitz, J. F. Heidelberg, T. Utterback, Y. H. Rogers, L. I Falcon, V. Souza, G. Bonilla-Rosso, L. E. Eguiarte, D. M. Karl, S. Sathyendranath, T. Platt, E. Bermingham, V. Gallardo, G. Tamayo-Castillo, M. R. Ferrari, R. L. Strausberg, K. Nealson, R. Friedman, M. Frazier, and J. C. Venter. The Sorcerer II Global Ocean Sampling expedition: Northwest Atlantic through eastern tropical Pacific. *PLoS Biol.*, 5:e77, 2007.

[213] Y. Sakakibara, M. Brown, R. Hughey, I. S. Mian, K. Sjölander, R. C. Underwood, and D. Haussler. Stochastic context-free grammars for tRNA modeling. *Nucl. Acids Res.*, 22:5112–5120, 1994.

[214] Y. Sakakibara, M. Brown, R. C. Underwood, I. S. Mian, and D. Haussler. Stochastic context-free grammars for modeling RNA. In Lawrence Hunter, editor, *Proceedings of the Twenty-Seventh Annual Hawaii International Conference on System Sciences: Biotechnology Computing*, volume V, pages 284–293, Los Alamitos, CA, 1994. IEEE Computer Society Press.

[215] A. Sandelin, W. Alkema, P. Engström, W. W. Wasserman, and B. Lenhard. JASPAR: an open-access database for eukaryotic transcription factor binding profiles. *Nucl. Acids Res.*, 32:D91–D94, 2004.

[216] D. Sankoff. Simultaneous solution of the RNA folding, alignment, and protosequence problems. *SIAM J. Appl. Math.*, 45:810–825, 1985.

[217] P. Schattner, S. Barberan-Soler, and T. M. Lowe. A computational screen for mammalian pseudouridylation guide H/ACA RNAs. *RNA*, 12:15–25, 2006.

[218] P. D. Schloss and J. Handelsman. Metagenomics for studying unculturable microorganisms: Cutting the Gordian knot. *Genome Biol.*, 6:229, 2005.

[219] M. E. Schmitt and D. A. Clayton. Nuclear RNase MRP is required for correct processing of pre-5.8S rRNA in *Saccharomyces cerevisiae*. *Mol. Cell. Biol.*, 13:7935–7941, 1993.

[220] P. Sellers. On the theory and computation of evolutionary distances. *SIAM J. Appl. Math.*, 26:787–793, 1974.

[221] M. Selmer, C. M. Dunham, , A. Weixlbaumer, S. Petry, A. C. Kelley, J. R. Weir, and V. Ramakrishnan. Structure of the 70s ribosome complexed with mRNA and tRNA. *Science*, 313:1935–1942, 2006.

[222] B. A. Shapiro and J. Navetta. A massively parallel genetic algorithm for RNA secondary structure prediction. *The Journal of Supercomputing*, 8:195–207, 1994.

[223] B. A. Shapiro, J. C. Wu, D. Bengali, and M. J. Potts. The massively parallel genetic algorithm for RNA folding: MIMD implementation and population variation. *Bioinformatics*, 17:137–148, 2001.

[224] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *J. Mol. Biol.*, 147:195–197, 1981.

[225] T. F. Smith, M. S. Waterman, and C. Burks. The statistical distribution of nucleic acid similarities. *Nucl. Acids Res.*, 13:645–656, 1985.

[226] J. Söding. Protein homology detection by HMM-HMM comparison. *Bioinformatics*, 21:951–960, 2005.

[227] D. Soll and U. L. RajBhandary. *tRNA: Structure, Biosynthesis, and Function*. ASM Press, Washington DC, 1995.

[228] G. A. Soukup and R. R. Breaker. Relationship between internucleotide linkage geometry and the stability of RNA. *RNA*, 5:1308–1325, 1999.

[229] A. K. Srivastava and D. Schlessinger. Structure and organization of ribosomal DNA. *Biochimie*, 73:631–638, 1991.

[230] M. R. Stahley and S. A. Strobel. RNA splicing: group I intron crystal structures reveal the basis of splice site selection and metal ion catalysis. *Curr. Opin. Struct. Biol.*, 16:319–326, 2006.

[231] M. Stanke and S. Waack. Gene prediction with a hidden Markov model and a new intron submodel. *Bioinformatics*, 19:ii215–ii225, 2003.

[232] P. Steffen, B. Voss, M. Rehmsmeier, J. Reeder, and R. Giegerich. RNAshapes: an integrated RNA analysis package based on abstract shapes. *Bioinformatics*, 22:500–503, 2006.

[233] T. A. Steitz and P. B. Moore. RNA, the first macromolecular catalyst: the ribosome is a ribozyme. *Trends Biochem. Sci.*, 28:411–418, 2003.

[234] T. A. Steitz and J. A. Steitz. A general two-metal-ion mechanism for catalytic RNA. *Proc. Natl. Acad. Sci. USA*, 90:6498–6502, 1993.

[235] G. D. Stormo. DNA binding sites: representation and discovery. *Bioinformatics*, 16:16–23, 2000.

[236] S. L. Stricklin. *Noncoding RNA Genes in* Caenorhabditis elegans. PhD thesis, Washington University School of Medicine, 2006.

[237] S. L. Stricklin, S. Griffiths-Jones, and S. R. Eddy. *C. elegans* noncoding RNA genes. In The *C. elegans* Research Community, editor, *WormBook*. doi/10.1895/wormbook.1.7.1, http://www.wormbook.org, 2005.

[238] V. S. Sunderam. PVM: a framework for parallel distributed computing. *Concurrency: Practice and Experience*, 2:315–339, 1990.

[239] C. A. Theimer and J. Feigon. Structure and function of telomerase RNA. *Curr. Opin. Struct. Biol.*, 16:307–318, 2006.

[240] J. D. Thompson, D. G. Higgins, and T. J. Gibson. CLUSTAL W: Improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties, and weight matrix choice. *Nucl. Acids Res.*, 22:4673–4680, 1994.

[241] N. Toor, K. S. Keating, S. D. Taylor, and A. M. Pyle. Crystal structure of a self-spliced group II intron. *Science*, 320:77–82, 2008.

[242] H. Touzet and O. Perriquet. CARNAC: folding families of related RNAs. *Nucl. Acids Res.*, 32:W142–W145, 2004.

[243] D. H. Turner, N. Sugimoto, J. A. Jaeger, C. E. Longfellow, S. M. Freier, and R. Kierzek. Improved parameters for prediction of RNA structure. *Cold Spring Harbor Symp. Quant. Biol.*, 52:123–133, 1987.

[244] J. C. Venter, K. Remington, J. F. Heidelberg, A. L. Halpern, D. Rusch, J. A. Eisen, D. Wu, I. Paulsen, K. E. Nelson, W. Nelson, D. E. Fouts, S. Levy, A. H. Knap, M. W. Lomas, K. Nealson, O. White, J. Peterson, J. Hoffman, R. Parsons, H. Baden-Tillson, C. Pfannkoch, Y. H. Rogers, and H. O. Smith. Environmental genome shotgun sequencing of the Sargasso Sea. *Science*, 304:66–74, 2004.

[245] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13:260–269, 1967.

[246] D. Vlieghe, A. Sandelin, P. J. De Bleser, K. Vleminckx, W. W. Wasserman, F. van Roy, and B. Lenhard. A new generation of JASPAR, the open-access repository for transcription factor binding site profiles. *Nucl. Acids Res.*, 34:D95–D97, 2006.

[247] C. Vogel, M. Bashton, N. D. Kerrison, C. Chothia, and S. A. Teichmann. Structure, function and evolution of multidomain proteins. *Curr. Opin. Struct. Biol.*, 14:208–216, 2004.

[248] R. Walczak, E. Westhof, P. Carbon, and A. Krol. A novel RNA structural motif in the selenocysteine insertion element of eukaryotic selenoprotein mRNAs. *RNA*, 2:367–379, 1996.

[249] I. M. Wallace, G. Blackshields, and D. G. Higgins. Multiple sequence alignments. *Curr. Opin. Struct. Biol.*, 15:261–266, 2005.

[250] A.E. Walter, D.H. Turner, J. Kim, M.H. Lyttle, P. Muller, D.H. Mathews, and M. Zuker. Coaxial stacking of helixes enhances binding of oligoribonucleotides and improves predictions of RNA folding. *Proc. Natl. Acad. Sci. USA*, 91:9218–9222, 1994.

[251] P. Walter and G. Blobel. Signal recognition particle contains a 7S RNA essential for protein translocation across the endoplasmic reticulum. *Nature*, 299:691–698, 1982.

[252] S. Washietl, I. L. Hofacker, and P. F. Stadler. Fast and reliable prediction of noncoding RNAs. *Proc. Natl. Acad. Sci. USA*, 102:2454–2459, 2005.

[253] K. M. Wassarman and G. Storz. 6S RNA regulates *E. coli* RNA polymerase activity. *Cell*, 101:613–623, 2000.

[254] Z. Weinberg and W. L. Ruzzo. Exploiting conserved structure for faster annotation of non-coding RNAs without loss of accuracy. *Bioinformatics*, 20 Suppl. 1:I334–I341, 2004.

[255] Z. Weinberg and W. L. Ruzzo. Faster genome annotation of non-coding RNA families without loss of accuracy. *RECOMB '04*, pages 243–251, 2004.

[256] Z. Weinberg and W. L. Ruzzo. Sequence-based heuristics for faster annotation of non-coding RNA families. *Bioinformatics*, 22:35–39, 2006.

[257] W. G. Weisburg, S. M. Barns, D. A. Pelletier, and D. J. Lane. 16S ribosomal DNA amplification for phylogenetic study. *J. Bacteriol.*, 173:697–703, 1991.

[258] S. Will, K. Reiche, I. L. Hofacker, P. F. Stadler, and R. Backofen. Inferring noncoding RNA families and classes by means of genome-scale structure-based clustering. *PLoS Comput. Biol.*, 3:e65, 2007.

[259] D. Wilson, R. Pethica, Y. Zhou, C. Talbot, C. Vogel, M. Madera, C. Chothia, and J. Gough. SUPERFAMILY–sophisticated comparative genomics, data mining, visualization and phylogeny. *Nucl. Acids Res.*, 37:D380–D386, 2009.

[260] R. Wilting, S. Schorling, B. C. Persson, and A. Böck. Selenoprotein synthesis in Archaea: identification of an mRNA element of *Methanococcus jannaschii* probably directing selenocysteine insertion. *J. Mol. Biol.*, 266:637–641, 1997.

[261] W. Winkler, A. Nahvi, and R. R. Breaker. Thiamine derivatives bind messenger RNAs directly to regulate bacterial gene expression. *Nature*, 419:952–956, 2002.

[262] W. C. Winkler and R. R. Breaker. Regulation of bacterial gene expression by riboswitches. *Annu. Rev. Microbiol.*, 59:487–517, 2005.

[263] W. C. Winkler, A. Nahvi, A. Roth, J. A. Collins, and R. R. Breaker. Control of gene expression by a natural metabolite-responsive ribozyme. *Nature*, 428:281–286, 2004.

[264] C. R. Woese and G. E. Fox. Phylogenetic structure of the prokaryotic domain: the primary kingdoms. *Proc. Natl. Acad. Sci. USA*, 74:5088–5090, 1977.

[265] A. Wozniak. Using video-oriented instructions to speed up sequence comparison. *Comput. Appl. Biosci.*, 13:145–150, 1997.

[266] Y. Yamaguchi, T. Maruyama, and A. Konagaya. High speed homology search with FPGAs. *Pac. Symp. Biocomput.*, pages 271–282, 2002.

[267] Z. Yao, Z. Weinberg, and W. L. Ruzzo. CMfinder–a covariance model based RNA motif finding algorithm. *Bioinformatics*, 22:445–452, 2006.

[268] D. H. Younger. Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10:189–208, 1967.

[269] M. Yousef, M. Nebozhyn, H. Shatkay, S. Kanterakis, L. C. Showe, and M. K. Showe. Combining multi-species genomic data for microRNA identification using a naive Bayes classifier. *Bioinformatics*, 22:1325–1334, 2006.

[270] Y.-K. Yu and T. Hwa. Statistical significance of probabilistic sequence alignment and related local hidden Markov models. *J. Comput. Biol.*, 8:249–282, 2001.

[271] Y. T. Yu, E. C. Scharl, C. M. Smith, and J. A. Steitz. The growing world of small nuclear ribonucleoproteins. In R. F. Gesteland, T. R. Cech, and J. F. Atkins, editors, *The RNA World, Second Edition*, pages 487–524. Cold Spring Harbor Laboratory Press, Cold Spring Harbor, New York, 1999.

[272] M. M. Yusupov, G. Z. Yusupova, A. Baucom, K. Lieberman, T. N. Earnest, J. H. Cate, and H. F. Noller. Crystal structure of the ribosome at 5.5 Å resolution. *Science*, 292:883–896, 2001.

[273] S. Zhang, B. Haas, E. Eskin, and V. Bafna. Searching genomes for noncoding RNA using FastR. *IEEE/ACM Trans. Comput. Biol. Bioinform.*, 2:366–379, 2005.

[274] Y. Zhang. Progress and challenges in protein structure prediction. *Curr. Opin. Struct. Biol.*, 18:342–348, 2008.

[275] F. Zinoni, J. Heider, and A. Böck. Features of the formate dehydrogenase mRNA necessary for decoding of the UGA codon as selenocysteine. *Proc. Natl. Acad. Sci. USA*, 87:4660–4664, 1990.

[276] M. Zuker. Computer prediction of RNA structure. *Meth. Enzymol.*, 180:262–288, 1989.

[277] M. Zuker. Mfold web server for nucleic acid folding and hybridization prediction. *Nucl. Acids Res.*, 31:3406–3415, 2003.

[278] M. Zuker and P. Stiegler. Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucl. Acids Res.*, 9:133–148, 1981.